1415 N CHERRY AVE
CHICAGO, IL 60642
(312) 281-6900
DMDII.ORG
DMDII@UILABS.ORG

DMDII
a UI LABS Collaboration
DIGITIZING AMERICAN MANUFACTURING

DMDII FINAL PROJECT REPORT

Integrated Manufacturing Variation Management (IMVM)	
Principle Investigator / Email Address	**Yujie Chen/chen_yujie@cat.com**
Project Team Lead	**Caterpillar Inc.**
Project Designation	**DMDII-14-07-02**
UI LABS Contract Number	**0220160004**
Project Participants	**The Board of Trustees of the University of Illinois - Urbana-Champaign** **Missouri University of Science and Technology**
DMDII Funding Value	**$793,327**
Project Team Cost Share	**$793,332**
Award Date	**January 19, 2016**
Completion Date	**July 31, 2018**

This project was completed under the Cooperative Agreement W31P4Q-14-2-0001, between U.S. Army - Army Contracting Command - Redstone and UI LABS on behalf of the Digital Manufacturing and Design Innovation Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

TABLE OF CONTENTS

Contents

1. EXECUTIVE SUMMARY

1.1 Industry Problems

Incoming stock from casting and forging suppliers currently varies to the point that standard machine tools cannot adequately respond the existing material condition in the as-programmed state. The machine tools inability to dynamically respond to material stock variation results in broken tooling, scrap parts, increase man hour investment in trying to machine parts and can drive severe delays that impact the entire manufacturing value stream from foundries to OEMs. The manufacturing community has attempted to solve this problem through part probing in the machine tool, programming sub-routines in the controller, or through manual adjustments made by the machinist. The current approach has yielded a sub-optimal process that still requires significant human intervention and does not guarantee a conforming part.

1.2 Goals and Outcomes

The goal of the IMVM project was to generate an automated system by which a manufacturer can compensate for machine tool workspace (machine tool) errors induced due to part, fixture, tooling, or machine tool errors. An automated process would drive significant reductions in new part and new fixture setup times, and identifying parts delivered with significant variations, in turn minimizing machining requirement to bring the parts within specific parameters, reducing the number of scrapped parts, manual intervention and machining setup time. The innovations from IMVM Project are focused on improving the present state of technology, yielding significant process improvement directly impacting reliability on parts, reducing the time to determine whether a part can be brought within specifications by machining, or quickly identifying if a part needs to be scrapped or returned to the supplier before spending man hours machining a part in an attempt to bring it into required specifications. To address the goal, two major technologies have been developed and validated from this project. First one is machine tool Volumetric Error Compensation (VEC) technology to measure machine tool error, generate error model and compensate the error through machine compensation table. Second one is Adaptive Machining (AM) technology using compensated tool path to accommodate the geometric variations on the incoming parts.

From this project, DMDII industrial members will realize decreased process ramp up time, scraped parts, quick identification of parts variation, reduction of man hours used to machine parts that are out of specifications, quicker feedback to suppliers regarding significant variations in their stock, a reduction in the time needed to set up a fixture due to the front end identification of variations of the parts needed for an entire fixture and driving overall improvement in the manufacturing process by enabling automated identification of variations in parts.

2. PROJECT REVIEW

This project team provides two separate technologies, one to compensate for machine volumetric error, the other to accommodate for incoming parts variations. To compensate for machine volumetric error, the academic partners helped develop standard procedures to measure the machine tool positional information using a laser tracker and develop customized computational methods to general volumetric error model from this measured positional information. Then, this error model was interpreted into 25 compensation tables with 1000 interpolation points and compensated modeled errors through the machine controller. More than 80% machine volumetric error reduction has been accomplished with a production machine asset. To achieve accommodating incoming parts variations, both on/off-line scanning approaches and virtual gage software were developed to determine the material condition of the incoming part. With this tool, an objective decision whether to reject the incoming part or not can be made before even cutting any metal. Furthermore, with acceptable parts, the software was able to incorporate the optimized offset information with the existing program to achieve adaptive machining based on scanned geometric variations and minimized the risk of tooling interference.

3. KPI'S & METRICS

Section should minimally include the following:

Metric of machine error compensation	Baseline	Goal	Results	Validation Method
Time required for measurement	16 hours	Less than 6 hours	Less than 5.5 hours	Validated on 4-axis machine (4mx2.2mx2.5m)
Comp/uncomp mean residual improvement	-	More than 80%	85%	Validated on 4-axis machine (4mx2.2mx2.5m)
Difference from B5 test	-	Less than 20%	Less than 10%	Diagonal error comparison
Training time	24 hours +	Less than 16 hours	Less than 16 hours	Reviewed with technician

Metric of adaptive machining	Baseline	Goal	Results	Validation Method
Compensate locater variation	-	5mm	10mm	Production validated
Software training time	-	Less than 16 hours	Less than 16 hours	Reviewed with operator

Scanning & calculation time	-	Less than 5% of cycle time	Less than 5% of cycle time	Off-line scanning
Insufficient stock detection rate	-	90%	Nearly 100%	Lan and production validation
System cost	-	Less than $100k	Less than $100k	Hardware and engineering costs

4. TECHNOLOGY OUTCOMES

4.1 Machine tool error compensation

4.1.1 System Overview

The Volumetric Error Compensation (VEC) component of this project involved the validation and use of VEC techniques to measure and model error in a machine tool using a laser tracker and active target and generate detailed compensation tables that can be used to correct the modeled error and improve machine accuracy. Figure 1 shows a flow chart of the operational process for VEC implementation.

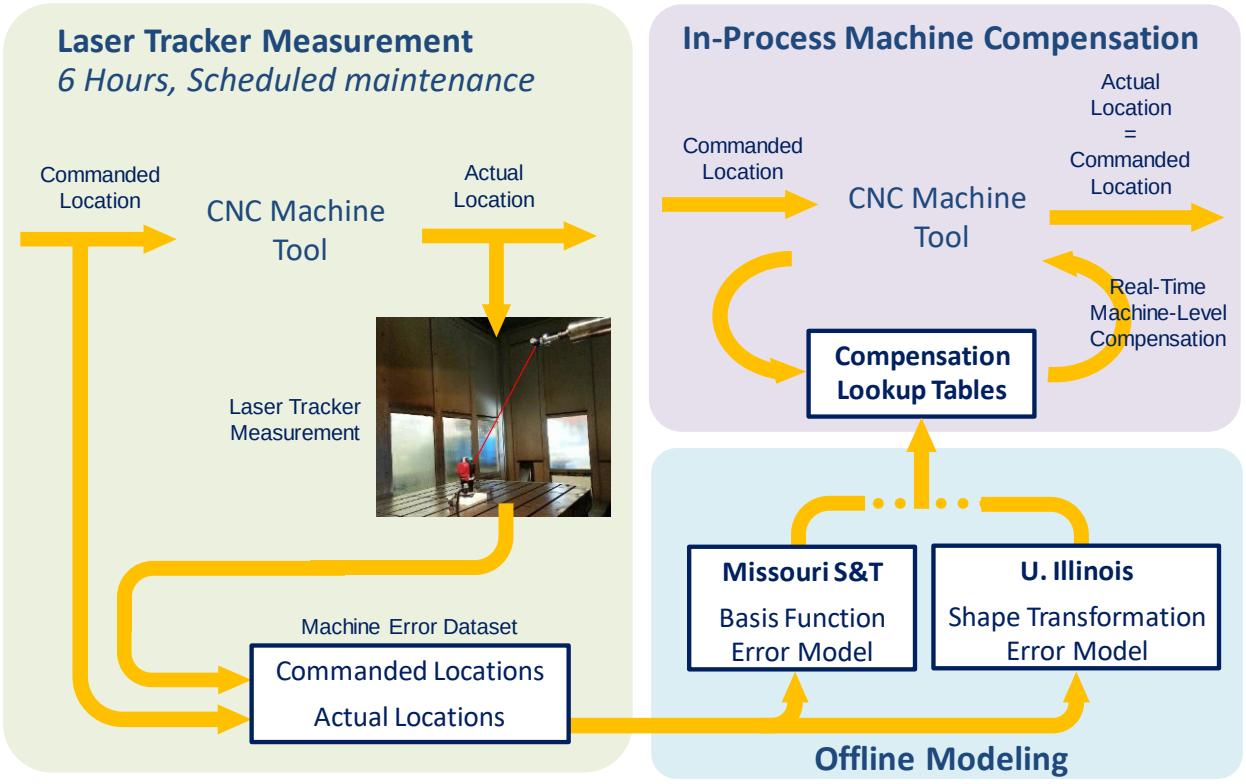


Figure 1 VEC System Overview

A system for grouping pseudo-random measurement points within the workspace was developed to prevent cable management problems. VEC modeling software was developed for the PAMA machine and several validation tests were run to demonstrate that at least 80% of the error in the uncompensated machine could be eliminated and that the VEC measurement collection process could be completed in under 6 hours.

To collect the VEC measurements, a laser tracker and active target are needed capable of automatically maintaining line of sight measurements as the machine is moved. For this project the API Radian laser tracker and API active target were used and it is assumed that equipment of similar capability will be used. The Spatial Analyzer (SA) metrology software package from New River Kinematics (NRK) was used to capture and export measurements from the laser tracker. The development done in this project assumes the use of this metrology software, but the general process could be adapted to any metrology system capable of collecting and exporting measurements from the laser tracker. The error modeling software developed for the PAMA robot requires a MATLAB software license. The compensation tables produced are 25 tables of 1000 points each. This is a greater number and size than most machine tools use by default, therefore it is necessary that the controller is set up to allow the use of these compensation table sizes and quantities.

4.1.2 System Architecture

The hardware system architecture for the measurement process is shown in Figure 2. The API Radian laser tracker was mounted in the center of the PAMA table and connected via its control box to a computer running the SA metrology software. The API Active Target is mounted on the wobble plate which is inserted into one of two tool shafts and mounted into the spindle of the machine. A photo of the actual hardware is shown in Figure 3.

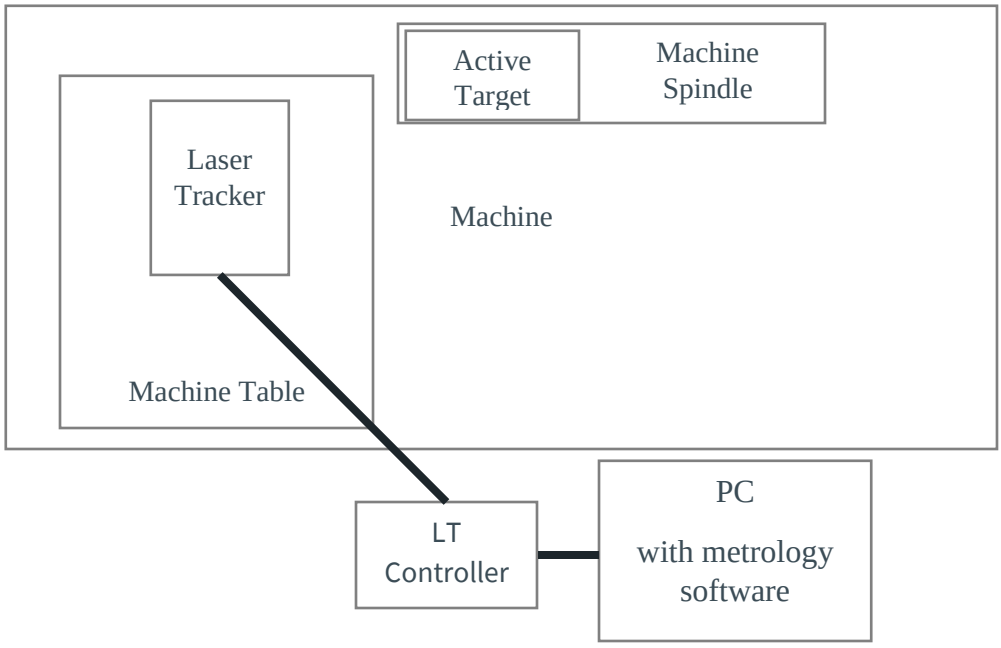


Figure 2: VEC measurement hardware architecture

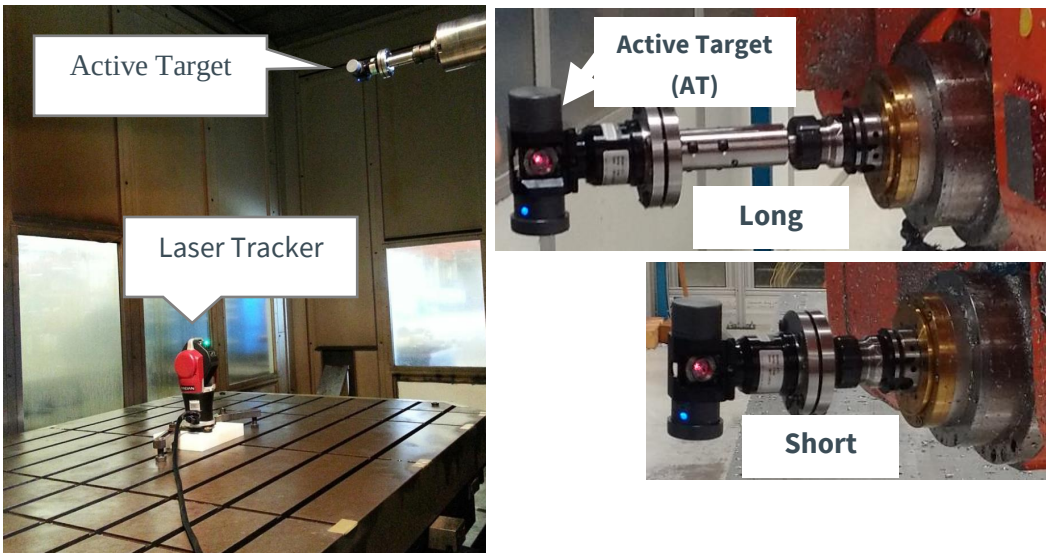


Figure 3: VEC measurement hardware setup

A flow chart of the software architecture showing input and outputs of each of the primary components is shown in Figure 4.

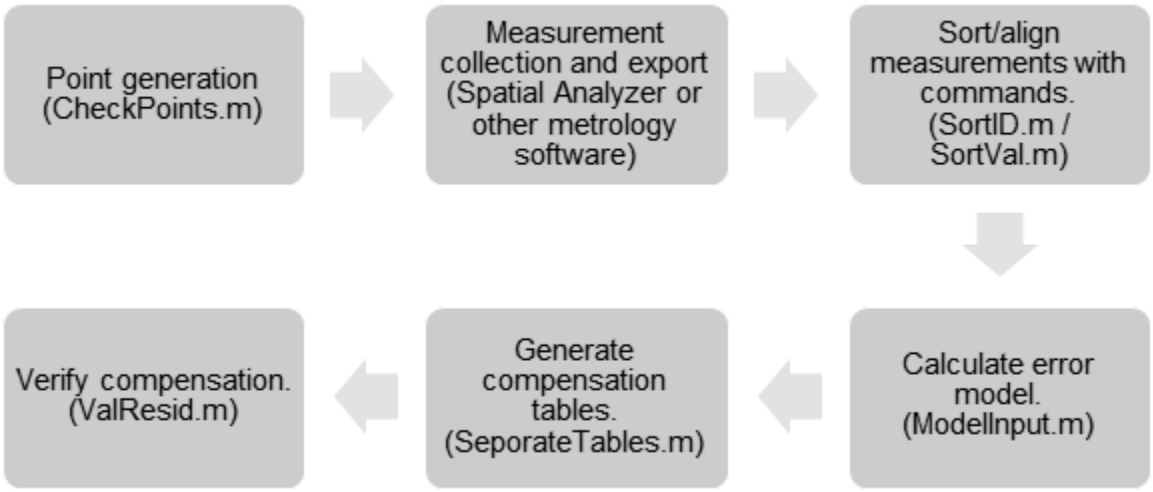


Figure 4 VEC software architecture

4.1.3 System Development

4.1.3.1 Modeling and Identification Methodologies

Nominal Kinematics

Nominal kinematic equations describe the orientation and position information of the tooltip with respect to the machine base frame. Given a set of axis commands, the nominal orientation and position information can be calculated by transforming from the base frame through a series

of coordinate frames that are set at each axis. Here, the Linear Homogeneous Transformation (LHT) is used between consecutive frames. The transformation from Frame *i*-1 to Frame *i* is

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} n_{i,x}^{i-1} & o_{i,x}^{i-1} & a_{i,x}^{i-1} & p_{i,x}^{i-1} \\ n_{i,y}^{i-1} & o_{i,y}^{i-1} & a_{i,y}^{i-1} & p_{i,y}^{i-1} \\ n_{i,z}^{i-1} & o_{i,z}^{i-1} & a_{i,z}^{i-1} & p_{i,z}^{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

where $\mathbf{T}_i^{i-1}$ is the transformation matrix from Frame *i*-1 to Frame *i*, $\mathbf{n}_i^{i-1} = \begin{bmatrix} n_{i,x}^{i-1} & n_{i,y}^{i-1} & n_{i,z}^{i-1} \end{bmatrix}^T$, $\mathbf{o}_i^{i-1} = \begin{bmatrix} o_{i,x}^{i-1} & o_{i,y}^{i-1} & o_{i,z}^{i-1} \end{bmatrix}^T$, $\mathbf{a}_i^{i-1} = \begin{bmatrix} a_{i,x}^{i-1} & a_{i,y}^{i-1} & a_{i,z}^{i-1} \end{bmatrix}^T$ and $\mathbf{p}_i^{i-1} = \begin{bmatrix} p_{i,x}^{i-1} & p_{i,y}^{i-1} & p_{i,z}^{i-1} \end{bmatrix}^T$ are the orientations and translations of the *x*, *y* and *z* axes of Frame *i* with respect to Frame *i*-1. The PAMA machine is a five-axis machine tool with four degrees of freedom (note the *W* and *Z* axes are in the same direction). A schematic of the manipulator is shown in Figure 5. The Active Target denotes the location where measurements are obtained.

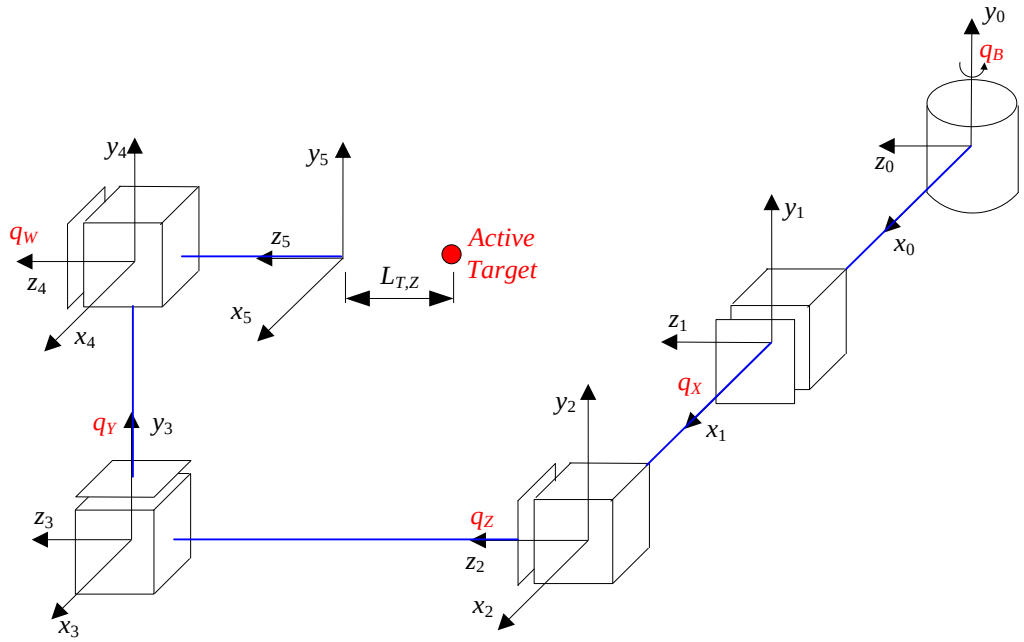


Figure 5: Schematic of kinematic structure with coordinate frames and kinematic model parameters labeled for PAMA machine tool.

Following the series of coordinate frames in Figure 5, the nominal forward kinematic model for the PAMA machine tool is

$$\mathbf{F}_\mathbf{n}(\mathbf{q}) = \mathbf{T}_1^0(q_B)\mathbf{T}_2^1(q_X)\mathbf{T}_3^2(q_Z)\mathbf{T}_4^3(q_Y)\mathbf{T}_5^4(q_W), \quad (2)$$

where $\mathbf{q} = [q_B, q_X, q_Z, q_Y, q_W]$ is the vector of axes commands and

$$\mathbf{T}_1^0(q_B)=\begin{bmatrix}\cos(q_B) & 0 & \sin(q_B) & 0\\ 0 & 1 & 0 & 0\\ -\sin(q_B) & 0 & \cos(q_B) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (3)$$

$$\mathbf{T}_2^1(q_X)=\begin{bmatrix}1 & 0 & 0 & q_X\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (4)$$

$$\mathbf{T}_3^2(q_Z)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & q_Z\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (5)$$

$$\mathbf{T}_4^3(q_Y)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & q_Y\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \quad (6)$$

$$\mathbf{T}_5^4(q_W)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & q_W\\ 0 & 0 & 0 & 1\end{bmatrix}. \quad (7)$$

Error Modeling

The method used to describe the machine tool geometric errors is called the Axis Perturbation (AP) Model Method, which is well-suited for generating machine tool compensation tables. This method regards the geometric errors as small position-dependent perturbations to the nominal axis commands. The model is described as

$$\mathbf{F}_{AP}(\mathbf{q})=\mathbf{F}_n\left(\mathbf{q}+\tilde{\mathbf{q}}(\mathbf{q})\right), \quad (8)$$

where $\tilde{\mathbf{q}}(\mathbf{q})=\begin{bmatrix}\tilde{q}_B(\mathbf{q}) & \tilde{q}_X(\mathbf{q}) & \tilde{q}_Z(\mathbf{q}) & \tilde{q}_Y(\mathbf{q}) & \tilde{q}_W(\mathbf{q})\end{bmatrix}$ is a vector of functions that perturb the nominal axis commands. The axis command perturbation functions are selected as an uncoupled sum of perturbations of each axis command (q_B, q_X, q_Z, q_Y, q_W) as,

$$\begin{aligned}
\tilde{q}_B(\mathbf{q}) &= f_{BB}(q_B) + f_{BX}(q_X) + f_{BZ}(q_Z) + f_{BY}(q_Y) + f_{BW}(q_W) \\
\tilde{q}_X(\mathbf{q}) &= f_{XB}(q_B) + f_{XX}(q_X) + f_{XZ}(q_Z) + f_{XY}(q_Y) + f_{XW}(q_W) \\
\tilde{q}_Z(\mathbf{q}) &= f_{ZB}(q_B) + f_{ZX}(q_X) + f_{ZZ}(q_Z) + f_{ZY}(q_Y) + f_{ZW}(q_W) \\
\tilde{q}_Y(\mathbf{q}) &= f_{YB}(q_B) + f_{YX}(q_X) + f_{YZ}(q_Z) + f_{YY}(q_Y) + f_{YW}(q_W) \\
\tilde{q}_W(\mathbf{q}) &= f_{WB}(q_B) + f_{WX}(q_X) + f_{WZ}(q_Z) + f_{WY}(q_Y) + f_{WW}(q_W)
\end{aligned} \quad , \qquad (9)$$

where $f_{ij}(q_j)$ is a scalar function mapping the axis command, q_j, on axis j onto a perturbation to the command for axis i.

To capture the position dependency of the error terms and describe the unknown functions $f_{ij}(q_j)$, a set of basis functions, which are orthogonal and have a similar scale over an interval [-1,1], are used. A set of polynomials that satisfy these two requirements are Chebyshev polynomials. A linear relationship is used to map the positive axis travel limit to 1 and the negative travel limit to -1. The Chebyshev polynomials are

$$\begin{aligned}
&c_0(\varphi) = 1,\ c_1(\varphi) = \varphi,\ c_2(\varphi) = 2\varphi^2 - 1,\ c_3(\varphi) = 4\varphi^3 - 3\varphi, \\
&c_4(\varphi) = 8\varphi^4 - 8\varphi^2 + 1, \ldots,\ c_{m+1}(\varphi) = 2\varphi c_m(\varphi) - c_{m-1}(\varphi)
\end{aligned} \quad , \qquad (10)$$

where m denotes the Chebyshev polynomial order. The sum of Chebyshev polynomials is

$$C(\varphi) = a_0 + a_1 c_1(\varphi) + a_2 c_2(\varphi) + \cdots + a_m c_m(\varphi). \qquad (11)$$

Then, the error terms in **Error! Reference source not found.** can be represented by m^{th} order Chebyshev polynomials as

$$f_{ij}(q_j) = a_{0ij} + a_{1ij} c_1(q_j) + a_{2ij} c_2(q_j) + \cdots + a_{mij} c_m(q_j), \qquad (12)$$

where i and j denotes the axes. In this framework, modeling of the error kinematics corresponds to selecting a sufficient order m and appropriate model coefficients, a_{0ij}, a_{1ij},…, a_{mij}.

Measurements of nominal and actual kinematic models

When taking measurements, an Active Target is rigidly attached to the machine tool spindle and measurements are taken with respect to a measurement frame which, in general, is not coincident with the machine tool's table base frame. Therefore, a transformation from this measurement frame to the machine tool base frame is needed. Incorporating the static translations from Frame 5 to the Active Target and the measurement frame to the machine tool base frame, the nominal and actual kinematic models, respectively, are

$$\mathbf{p}_n^m(q) = \mathbf{T}_0^m \mathbf{F}_n(q) \mathbf{p}_T^5, \quad (13)$$

$$\mathbf{p}_a^{m'}(\mathbf{q}) = \mathbf{T}_0^{m'} \mathbf{F}_{AP}(\mathbf{q}) \mathbf{p}_{T'}^5, \quad (14)$$

where $\mathbf{p}_n^m(\boldsymbol{\varphi})$ = $[x_n, y_n, z_n, 1]^T$ is the predicted nominal world space position vector of the Active Target consisting of the xyz coordinates of the tool in mm with respect to the nominal measurement frame, $\mathbf{p}_a^{m'}(\boldsymbol{\varphi})$ = $[x_a, y_a, z_a, 1]^T$ is the predicted actual world space position vector of the Active Target consisting of the xyz coordinates of the tool in mm with respect to the actual measurement frame, $\mathbf{T}_0^m$ is the transformation from the nominal measurement frame to the machine table base frame, $\mathbf{T}_0^{m'}$ is the transformation from the actual measurement frame to the machine table base frame, $\mathbf{p}_T^5$ is the translation from the nominal spindle tip position to the nominal Active Target position, $\mathbf{p}_{T'}^5$ is the translation from the nominal spindle tip position to the actual Active Target position. As the actual measurement frame is arbitrarily constructed and placed in the measurement software, there is an error transformation between the actual measurement frame and the nominal measurement frame. Thus, $\mathbf{T}_0^{m'}$ contains two components: a nominal component and an error component. That is,

$$\mathbf{T}_0^{m'} = \mathbf{T}_0^m \mathbf{E}_m^{m'}, \quad (15)$$

where $\mathbf{E}_m^{m'}$ is a static error transformation from the actual measurement frame to the nominal measurement frame

$$\mathbf{E}_m^{m'} = \begin{bmatrix} 1 & -\varepsilon_{Z,0} & \varepsilon_{Y,0} & \delta_{X,0} \\ \varepsilon_{Z,0} & 1 & -\varepsilon_{X,0} & \delta_{Y,0} \\ -\varepsilon_{Y,0} & \varepsilon_{X,0} & 1 & \delta_{Z,0} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (16)$$

where $\varepsilon_{i,0}$ is a rotation about the i^{th} axis (rad) of the actual measurement frame, and $\delta_{i,0}$ is a translation along the i^{th} axis (mm) of the actual measurement frame. Through an appropriate choice in the measurement software, the actual measurement frame can be put close to the machine table base frame such that the nominal measurement frame and the machine tool base frame is the same and $\mathbf{T}_0^m = \mathbf{I}_4$, where $\mathbf{I}_4$ is a 4×4 identity matrix. Since the Active Target is rigidly attached to the spindle, the translation from the spindle tip position to the measurement point is

$$\mathbf{p}_{T'}^5 = \mathbf{p}_T^5 + \mathbf{p}_{T'}^T, \quad (17)$$

where $\mathbf{p}_{T'}^{T}$ is a static error translation from the nominal target point to the actual target point. Since the tool direction is along the q_W direction, the tool length nominal and error translation only have an element along the z_5 axis,

$$\mathbf{p}_{T'}^{T} = \begin{bmatrix} 0 & 0 & \delta_{T,Z} & 0 \end{bmatrix}^{T}, \tag{18}$$

and

$$\mathbf{p}_{T}^{5} = \begin{bmatrix} 0 & 0 & L_{T,Z} & 1 \end{bmatrix}, \tag{19}$$

where $\delta_{T,Z}$ (mm) is a translational error to correct the nominal tool length component $L_{T,Z}$ (mm).

To evaluate the results of modeling performance, the nominal residual, i.e., the error between the predicted nominal position and actual measured position is

$$\mathbf{r}_{j}^{n} = \mathbf{p}_{n}^{m'}\left(\mathbf{q}_{j}\right) - \mathbf{m}_{j}, \tag{20}$$

where $\mathbf{m}_j$ = $[x_m, y_m, z_m, 1]^T$ is the measurement vector corresponding to the command set $\mathbf{q}_j$ composed of the measured xyz positions of the active target in mm. Similarly, the model residual, i.e., the error between the predicted modeled position and the actual measured position, is

$$\mathbf{r}_{j}^{a} = \mathbf{p}_{a}^{m'}\left(\mathbf{q}_{j}\right) - \mathbf{m}_{j}. \tag{21}$$

The magnitude of the nominal and actual residuals, respectively, are found using the Euclidean norm,

$$\begin{aligned} r_{j}^{n} &= \sqrt{\left(x_{p,n} - x_{m,j}\right)^{2} + \left(y_{p,n} - y_{m,j}\right)^{2} + \left(z_{p,n} - z_{m,j}\right)^{2}} = \left\|\mathbf{r}_{j}^{n}\right\|_{2} \\ r_{j}^{a} &= \sqrt{\left(x_{p,a} - x_{m,j}\right)^{2} + \left(y_{p,a} - y_{m,j}\right)^{2} + \left(z_{p,a} - z_{m,j}\right)^{2}} = \left\|\mathbf{r}_{j}^{a}\right\|_{2} \end{aligned}. \tag{22}$$

Identification

Once the complete actual measurement model has been derived as in **Error! Reference source not found.**, the task becomes to determine the Chebyshev polynomials coefficients for each error term. The polynomial order should be large enough to capture all of the dominant error characteristics, but not too large as to over fit the data. The determination of

the coefficients is done using a maximum likelihood estimator as described in [1]. Refer to [1] for details regarding the method.

4.1.3.2 Mechanical Accuracy and Repeatability

Measurement Device and Accuracy

Measurement accuracy refers to how accurate the laser tracker can collect measurements. Usually, it is expressed by the standard deviation from the normal distribution of measurement errors caused by the laser tracker. According to API Radian specifications, the static measurement accuracy A_s=5 ppm (2σ). The measurement standard deviation is

$$\sigma_m = \frac{A_s L}{2}, \tag{23}$$

where A_s is the static measurement accuracy (ppm) of the Laser Tracker and L is the characteristic distance between the laser tracker and the Active Target (m). At the Caterpillar Technical Center, the characteristic distance was about L=3 m. This leads to a computed measurement standard deviation of $\sigma_m = 7.5\ \mathrm{um}$.

Mechanical Repeatability

Mechanical repeatability is a measure of how close the tooltip can return to a specific position. Bidirectional mechanical repeatability, which is the measure of a machine tool's ability to return to a specific position regardless of the approach direction, is representative in describing the machine tool geometric performance.

The process for calculating the bidirectional mechanical repeatability is as follows. First, n measurements are collected for m points. For the i^{th} point, the mean in the x- y-, and z-directions, respectively, is

$$\overline{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}, \tag{24}$$

$$\overline{y}_i = \frac{1}{n}\sum_{j=1}^{n} y_{i,j}, \tag{25}$$

$$\overline{z}_i = \frac{1}{n}\sum_{j=1}^{n} z_{i,j}. \tag{26}$$

where $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$ are the x, y, and z components, respectively, of the j^{th} measurement of the i^{th} point. The Euclidian value of the error between the j^{th} measurement and the centroid of the i^{th} point is

$$e_{i,j} = \sqrt{\left(x_{i,j} - \bar{x}_i\right)^2 + \left(y_{i,j} - \bar{y}_i\right)^2 + \left(z_{i,j} - \bar{z}_i\right)^2}. \quad (27)$$

The mean error, e_{mean}, of all the observed errors is the mean mechanical repeatability and the largest observed error, e_{max}, is the maximum mechanical repeatability.

$$e_{mean} = \frac{1}{mn}\sum_{i}^{m}\sum_{j}^{n} e_{i,j}, \quad (28)$$

$$e_{max} = \max\left(e_{i,j}\right), \quad (29)$$

Also, all the errors are fitted with a gamma distribution. Under the fitted gamma distribution, the value under which the area covers 99% of the entire distribution is named as the Gamma error

$$e_{Gamma} = e_{<99\%}. \quad (30)$$

For the Caterpillar event, 10 measurements were collected for 8 positions. Figure 6 shows the nominal plot of all the errors and the corresponding definitions for mean error, maximum error and Gamma error. Table 1 gives the error values. The Gamma error is about 0.02 mm which indicates that with a probability of 99.0% that repeated measurements at the same point will stay within in a ±0.02 mm range.

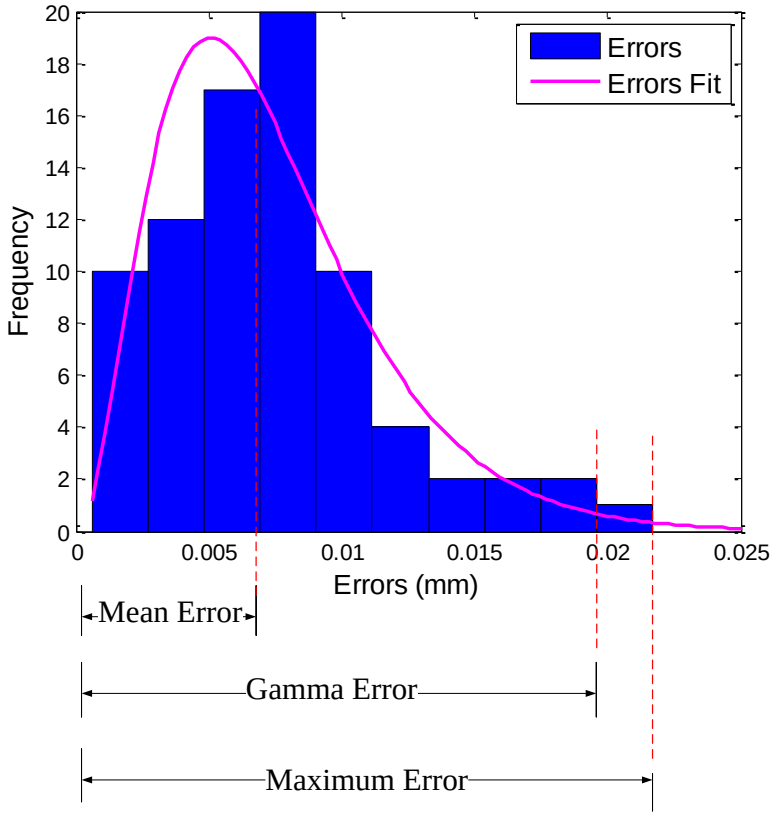


Figure 6: Fit of all errors with mean, maximum and Gamma errors.

Table 1: Identified mechanical repeatability of Caterpillar's PAMA machine tool.

	Mean Error (mm)	Maximum Error (mm)	Gamma Error (mm)	Number of Points	Number of Cycles
Repeatability	0.0074	0.0217	0.0200	8	10

Command Repeatability

Unlike mechanical repeatability which is the combination of the motion of all axes, command repeatability refers to how repeatable each individual axis is. It is often expressed by a standard deviation in commanded positioning error which is calculated by actuating each axis individually and taking measurements of a single joint command position as approached from a variety of distances. For each of the five axes, ten measurements (five from each direction) were taken at one point near the center of the machine's workspace. The nominal motion is rotary for the B axis and linear for other axes.

For the linear axes, the mean of the ten measurements are calculated. Since the linear motion is along one axis, only one axis (*x*, *y* or *z*) measurements information is needed. Taking the *x* axis as example, the mean of the measurements along the *x*- direction is calculated,

$$p_{X,a} = \frac{1}{10}\sum_{i=1}^{10} p_{X,i}\ , \tag{31}$$

where $p_{X,i}$ is ith measurement along the *x*- direction and $p_{X,a}$ is the mean value. Then, the linear error $\gamma_{X,i}$ (mm) between the i^{th} measurement and the mean value is calculated,

$$\gamma_{X,i} = p_{X,i} - p_{X,a}\,. \tag{32}$$

For the other axes, *Z*, *Y* and *W*, the same procedure as the *X* axis is followed and the linear errors, $\gamma_{Z,i}$, $\gamma_{Y,i}$ and $\gamma_{W,i}$, are calculated.

For the *B* axis, a circle can be fit to single rotation measurements. This motion is also assumed to be planar and only the *x*- and *z*-direction components of the measurements are analyzed.

$$x_{B,a} = \frac{1}{10}\sum_{i=1}^{10} x_{B,i}\ , \tag{33}$$

$$z_{B,a} = \frac{1}{10}\sum_{i=1}^{10} z_{B,i}\ , \tag{34}$$

where $\left(x_{B,i}, z_{B,i}\right)$ is the i^{th} measurement (mm). Letting the fitted circle center point be $\left(x_{B,c}, z_{B,c}\right)$, the angular error $\beta_{B,i}$ (deg) between the i^{th} measurement and the centroid $\left(x_{B,a}, z_{B,a}\right)$ with respect to the circle center point is

$$\beta_{B,i} = \arctan\left(\frac{z_{B,i} - z_{B,c}}{x_{B,i} - x_{B,c}}\right) - \arctan\left(\frac{z_{B,a} - z_{B,c}}{x_{B,a} - x_{B,c}}\right). \tag{35}$$

After the linear and rotation errors are calculated for each axis, nominal plots are fit for each single axis errors. The command standard deviations are also identified. Figure 7 shows the normal distribution of angular and linear errors for each axis and Table 2 lists the identified command standard deviations from the normal distributions.

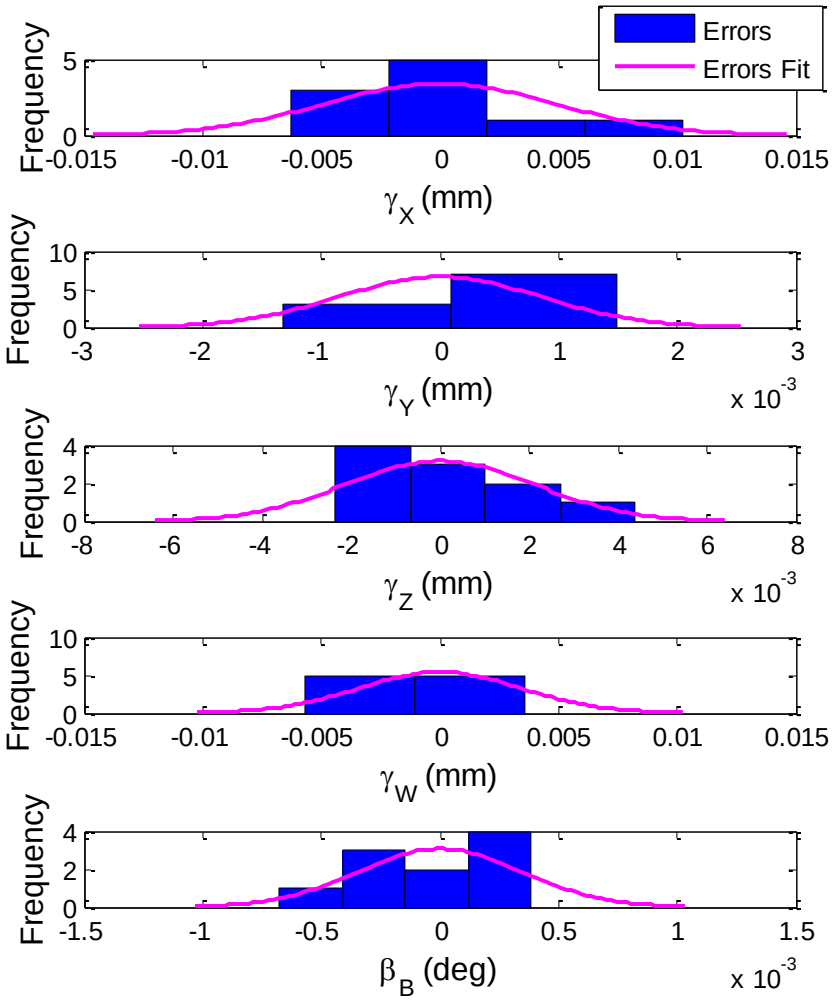


Figure 7: Normal distributions of errors of each axis.

Table 2: Identified command standard deviation of each axis.

Axis	Command Standard Deviation
X	4.9 um
Y	0.8 um
Z	2.1 um
W	3.4 um
B	0.3×10^{-3} degree

4.1.3.3 Thermal Effects

In addition to considering the short-term mechanical repeatabilities, the long-term thermal repeatability of both the machine and the laser tracker were evaluated.

Hard Points

During the PAMA calibration in July 2016, two measurements locations, referred to as hard points (HP), were setup on the table as shown in Figure 8. These two locations were measured continuously throughout the night between the second and third day of the calibration event as well as various times throughout the third day between collecting measurement sets.

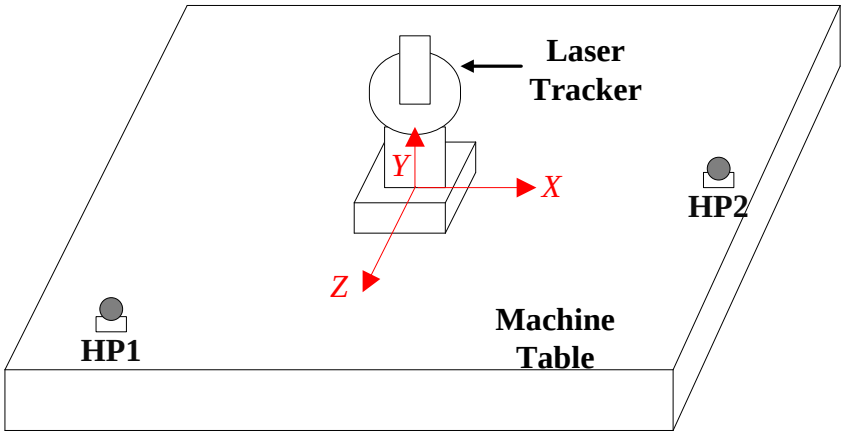


Figure 8: The schematic of the machine table, the placement of laser tracker and three hard points.

The measurements of these hard points were then used to determine if any expansion of the machine table or deformation laser tracker occurred. In order to get an idea of the amount of expansion witnessed in the table, the distance between the two hard points was calculated. The plot of these distances versus temperature is shown below in Figure 9.

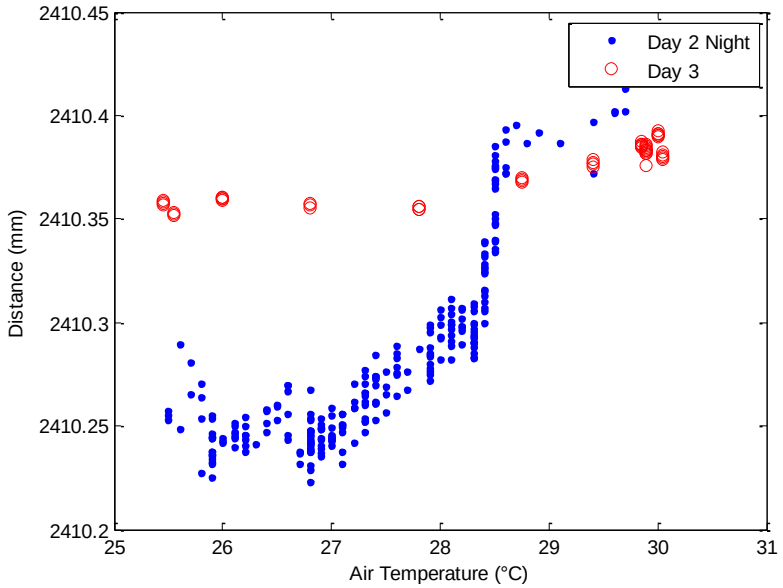


Figure 9: Distance between hard points 1 and 2.

While Figure 9 shows a positive relationship between air temperature and distance between hard points 1 and 2 for the night measurements, this relationship is clearly nonlinear. This is most likely due to the discrepancy between the air temperature and the machine temperature. **In future thermal deformation measurements of the machine tool, measurement of the machine temperature, rather than air temperature, is recommended**.

In addition to examining expansion in the table itself, expansion in the vertical direction (perpendicular to the table), which are most likely attributed to either the tracker or the tracker mounting block, were also considered. A plot of these measurements is shown below in Figure 10.

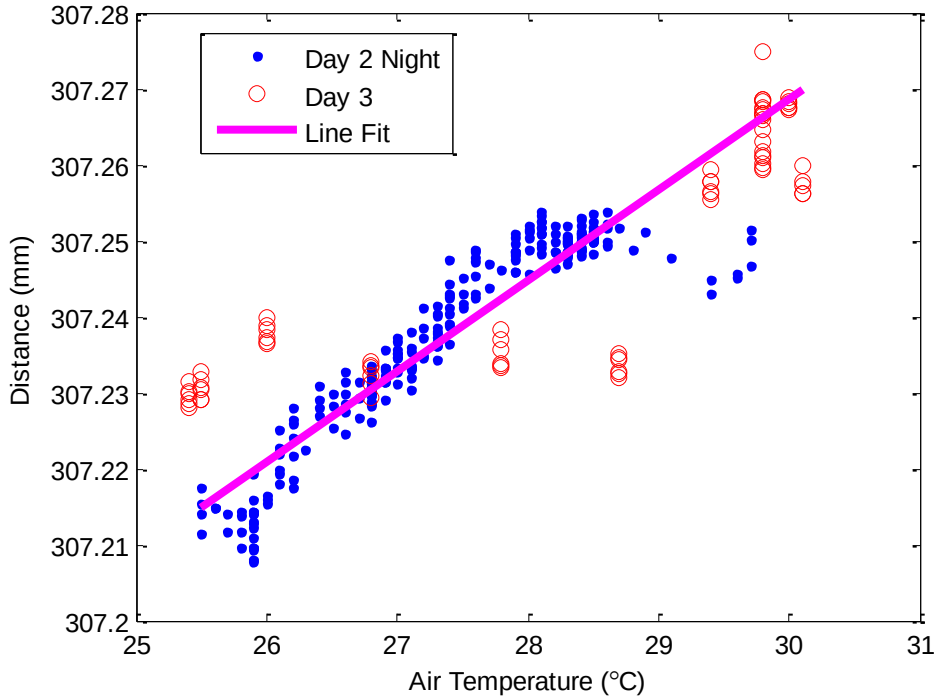


Figure 10: Vertical shift of tracker\mounting block.

From the data in Figure 10, there appears to be linear relationship between the vertical expansion and the air temperature that appears to have a fairly constant slope of 0.012 mm/°C. These results indicate that **the laser tracker experienced thermal expansion with good correlation to air temperature**.

B5 Test

In order to better understand the effects of thermal change on the machine as a whole, additional data from the B5 test implemented by Caterpillar on 07/14/2016 was examined. To better capture thermal effects on the machine, the standard B5 test was modified to last longer in order to incorporate temperature varying effects. Similar to a normal B5 test, measurements were taken at 11 positions on the W axis with respect to the table position as shown in Table 3.

Table 3: Commanded 11 positions on W axis for B5 test.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Command (mm)	-50	-120	-190	-260	-330	-400	-470	-540	-610	-680	-750

In each cycle of the test, the full sequence of 11 points was measured forward and backward such that each position was measured twice. After each cycle, approximately 10 minutes was waited before beginning the next cycle. The cycle was repeated a total of 11 times which was taken over about a 1.5 hours period (about 2˚C machine temperature change). With the data from modified test, Figure 11 shows the linear errors at different positions under different machine temperatures.

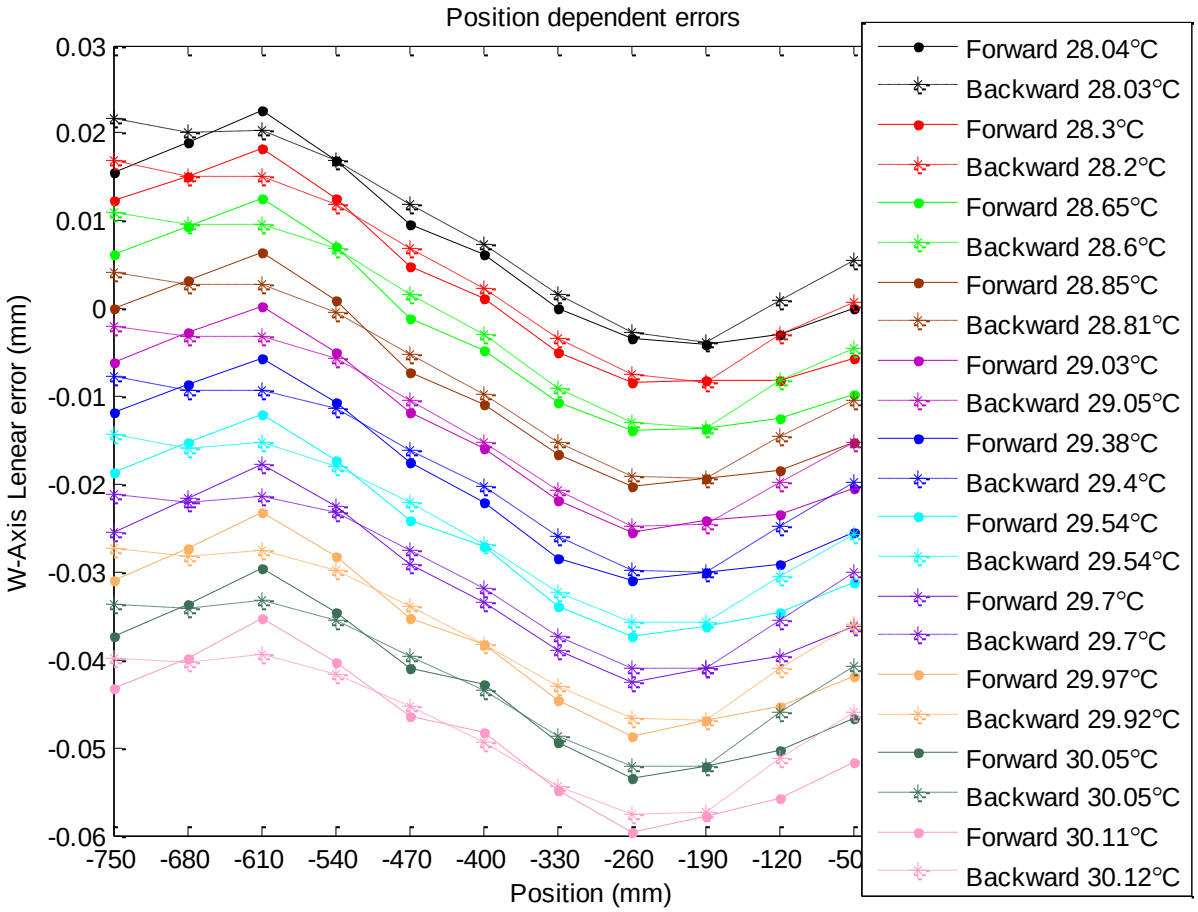


Figure 11: Linear errors along W axis with respect to positions from B5 test.

Figure 11 shows the position errors as would typically be displayed in the B5 test, although the cycles are typically more closely grouped because there is less delay, and hence less temperature change, between them. This figure shows a **clear offset in the linear errors at each different machine temperature**. To better illustrate this point, the linear error was instead plotted against machine temperature for each measurement position as shown below in Figure 12.

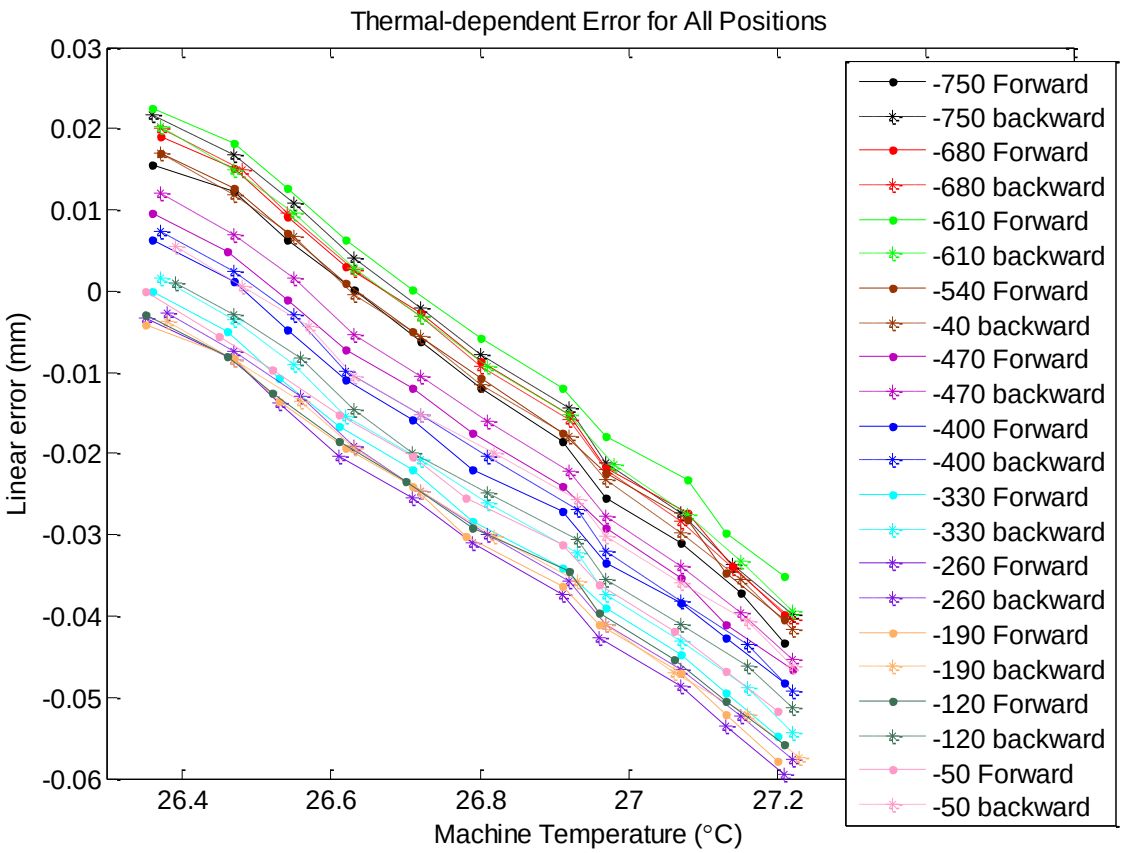


Figure 12: Linear errors along W axis with respect to machine temperature from B5 test.

Figure 12 shows a **clear linear trend between linear error and machine temperature**. The slope of the linear error with respect to machine temperature as well as the slope of the straightness errors with respect to temperature is shown below in Table 4.

Table 4: Rates of linear, straightness error change over machine temperature change from B5 test.

Linear Error (W) (mm/˚C)	**Straightness Error (XX) (mm/˚C)**	**Straightness Error (YY) (mm/˚C)**	**Magnitude Error (mm/˚C)**
0.060	0.021	0.0015	0.064

From the thermal analysis of both the hard point measurements and the B5 data, there is clear evidence that **both the machine and the laser tracker experienced thermal expansion**. This fact should be considered while examining the modeling and validation results shown in the following sections.

Feb 2017 Hard Points

To analyze thermal variance in the tracker and table, measurements of 6 SMRs placed on the surface of the table and one in the tracker 'birdbath' were taken overnight. Analysis of the July 2016 event data indicated that measurements taken in absolute distance mode (ADM) showed

a radial drift which seemed to have a thermal component, but that interferometer mode (IFM) measurements did not show significant drift. IFM measurements require continuous visibility of the reflector to the tracker and any time there is any interference blocking the beam ADM measurements are used to initialize further IFM measurements. To correct for the drift, an ADM measurement of a known location such as the tracker 'birdbath' can be measured in ADM mode to determine the radial drift. For the overnight measurements the radial distance to each SMR is shown in Figure 13. Figure 14 shows the temperature recorded by the tracker.

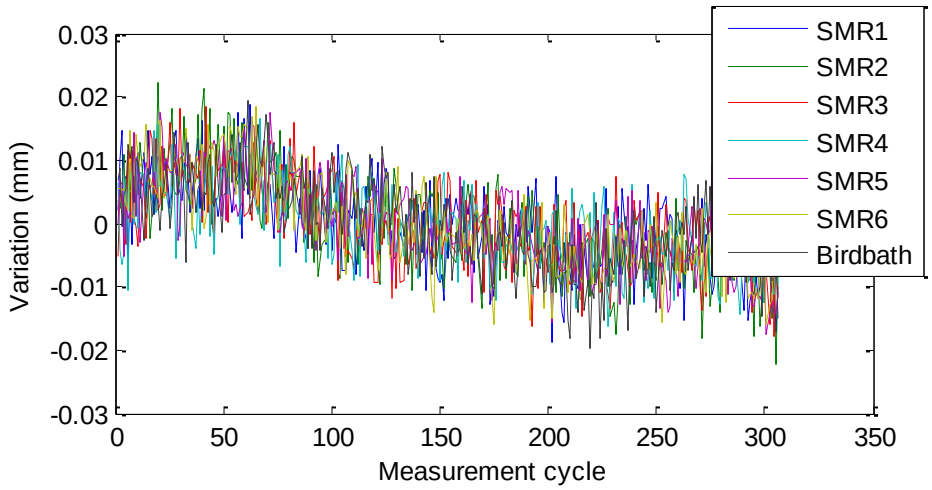


Figure 13: Radial drift overnight.

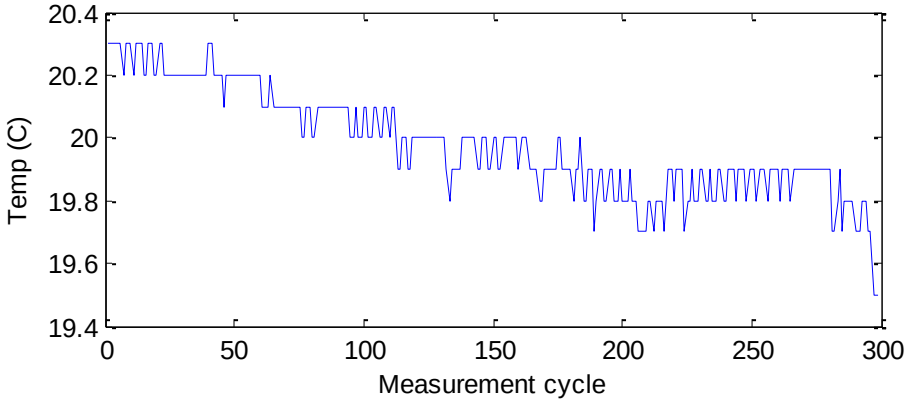


Figure 14: Temperature overnight.

While there is some visible drift, the total drift observed is of similar magnitude as the measurement noise. The temperature variance was less than 1 ºC

Thermal Measurements During Operation

Thermal data was collected during the measurement of the identification sets during the February 2017 testing using the sensors placed for the thermal tests covered in section 4.2. Figure 15 shows the measurements for the sensors connected to channels 3,4,5 and 7. Channel 6 was disconnected and channels 1,2, and 8 were much noisier and are shown in Figure 16

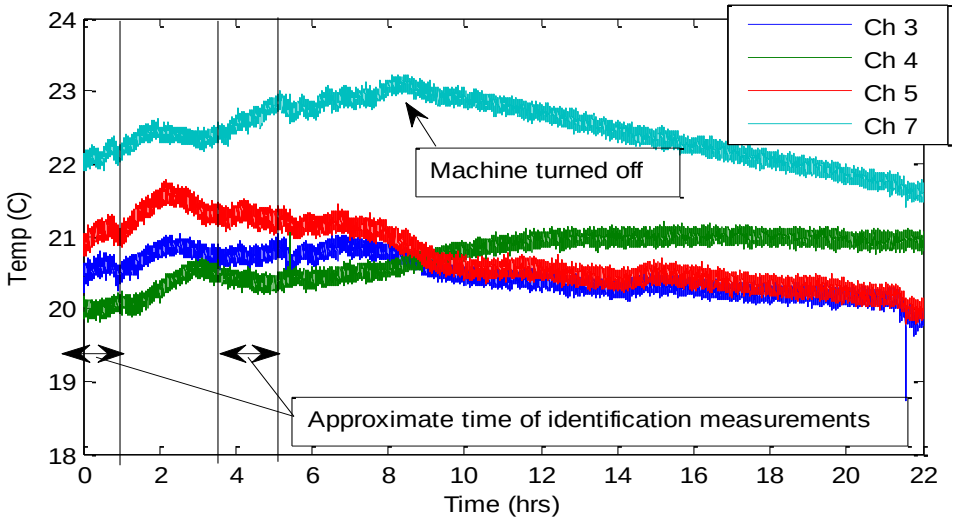


Figure 15: Thermal measurements from the day identification measurements were done

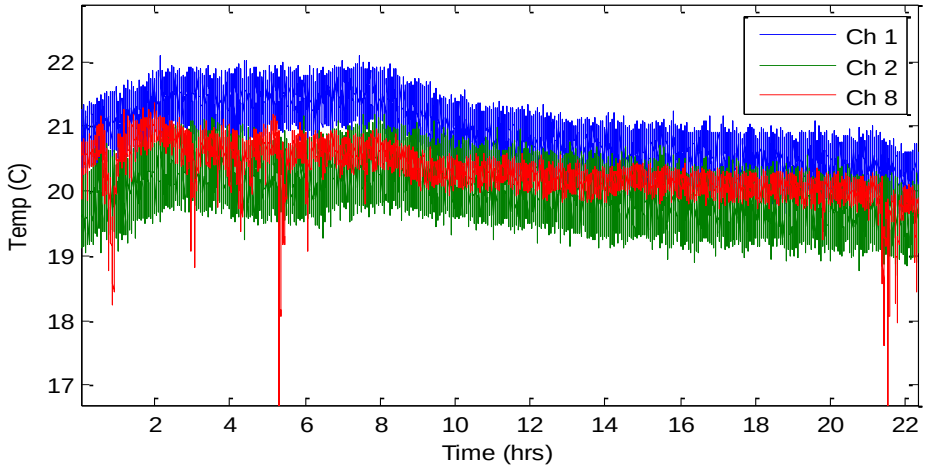


Figure 16: Thermal measurements from the day identification measurements were done.

The largest temperature variations changed by around 1 ºC across the identification measurements.

During the measurement of the identification sets, any time the beam was interrupted an ADM mode measurement of the tracker birdbath was taken to check for radial drift of the ADM measurements. No variance greater than the expected measurement noise was observed, so it was determined that the identification measurements did not require radial adjustment to compensate for thermal drift of the ADM measurements during that event.

4.1.3.4 Tool Length Difference Observations

To test the consistency between the two sets of identification data and investigate the thermal influence, the distance between the long and short tool measurements was calculated for each

point. If the measurements are consistent and there are no changes in the measurement device or machine, the variation in these distances should be equal to the mechanical repeatability of the machine. During the July 2016 event, large variations in tool length were observed and this was attributed to thermal variation caused by a 3.7 ºC temperature change. During this event the temperature change was only observed to be around 1 ºC. Since the temperature change was much lower, then any inconsistencies due to temperature would also be expected to be smaller. In order to compare these distances to the mechanical repeatability, the following calculation was performed. Let $\begin{bmatrix} x_{s,i} & y_{s,i} & z_{s,i} \end{bmatrix}^T$ be the measurement of the i^{th} point with the short tool and $\begin{bmatrix} x_{l,i} & y_{l,i} & z_{l,i} \end{bmatrix}^T$ be the measurement of the i^{th} point with the long tool, the average distance between the short tool and long tool length is

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(x_{s,i}-x_{l,i}\right)^2+\left(y_{s,i}-y_{l,i}\right)^2+\left(z_{s,i}-z_{l,i}\right)^2}\,. \tag{35}$$

Thus, the distance error between the short tool and long tool for the i^{th} point is

$$d_i = \sqrt{\left(x_{s,i}-x_{l,i}\right)^2+\left(y_{s,i}-y_{l,i}\right)^2+\left(z_{s,i}-z_{l,i}\right)^2}-\bar{d}\,. \tag{36}$$

A histogram of these distances as originally observed is shown below in Figure 17.

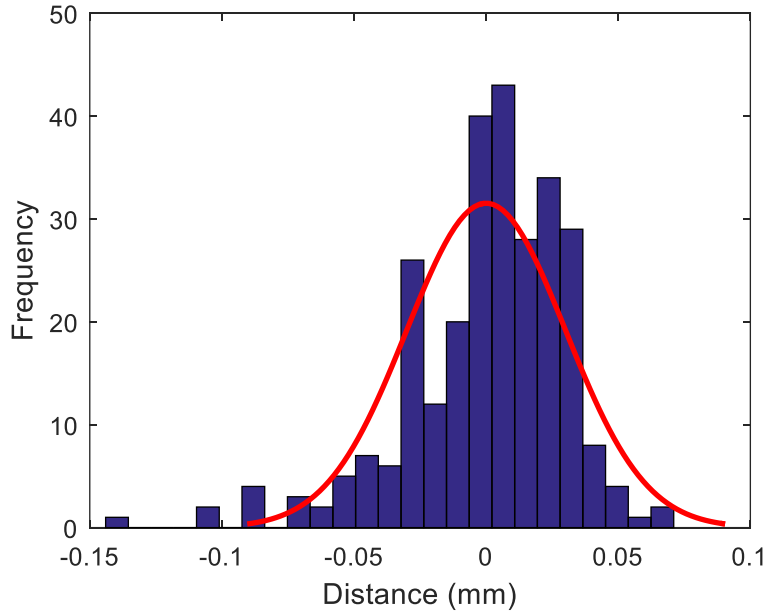


Figure 17: The initial observed variation of distance between short tool and long tool measurements.

In Figure 17, the distance between measurements with the two different tools has a spread of 0.207 mm. This value is around ten times the mechanical repeatability which indicates that there is clear inconsistency between the two measurement sets on a similar or greater order of magnitude as the July 2016 measurement. From the previous thermal repeatability analysis, the observed low temperature variation would not be expected to produce this level of error.

Several custom tests were run to isolate the root cause of the persistent error since it did not appear to be explainable by thermal variation. The tool length errors were found to correlate highly with the incidence angle between the laser and the tool. This incidence angle is illustrated in Figure 18 and the tool length errors are plotted with respect to incidence angle in Figure 19.

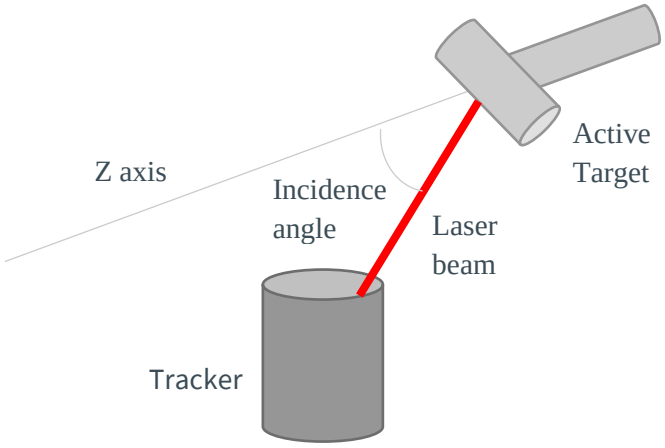


Figure 18: Illustration of incidence angle.

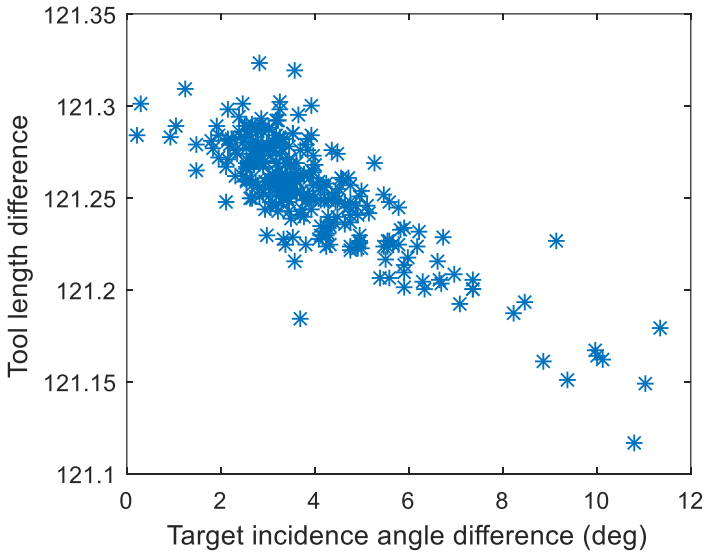


Figure 19: Tool length differences with respect to incidence angle as originally measured

The cause of this trend was discovered to be that the ADM offset for the active target set in the measurement software had been changed to an incorrect value sometime prior to the July measurement event. This effectively caused all measurements to be shifted in the radial direction

0.9617 mm away from the tracker. After adjusting the data points to reverse this effect, the trend with respect to incidence angle disappeared and the tool length errors were reduced. Figure 20 and Figure 21 show the tool length error histogram and plotted with respect to incidence angle after correction.

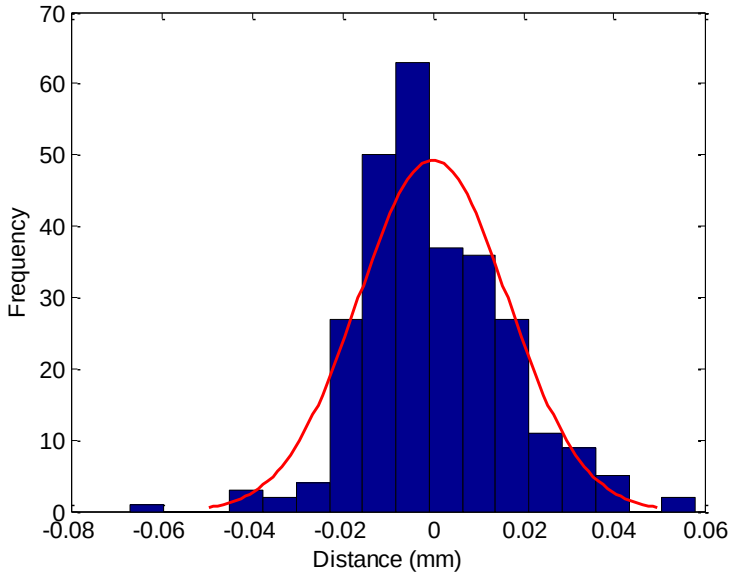


Figure 20: Tool length difference histogram after correcting for the erroneous active target ADM offset. Remaining errors are a combination of measurement repeatability plus mechanical repeatability plus thermal deformation effects.

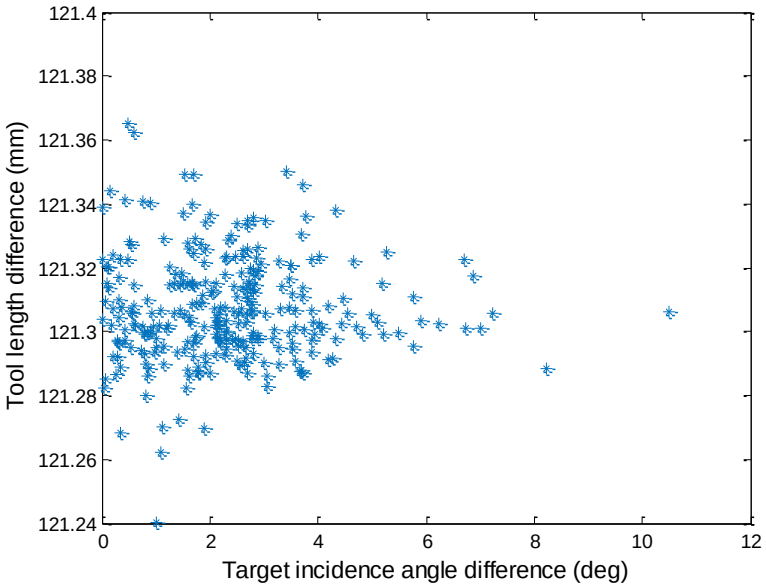


Figure 21: Tool length differences with respect to incidence angle after active target ADM offset correction

4.1.3.5 Modeling Implementation and Compensation

Model Parameter Standard Deviation

During the identification process, due to the property of identification algorithm, a set of model parameter standard deviation should be used to determine the model parameters (i.e., the coefficients of the Chebyshev polynomials). In the identification process, the model parameters are assumed to have zero mean and normal distribution and behave as tuning parameters. Through multiple tuning attempts, the standard deviations used in identifying the modeling error parameters for PAMA machine tool are determined as listed in Table 5.

Table 5: Model parameter standard deviations for each joint error model.

Frame	Rotation (rad)	Translation (mm)
Base	0.1	5
B Axis	0.005	
X Axis	0.05	
Z Axis	0.05	
Y Axis	0.05	
W Axis	0.05	

Measurements

A picture of the machine tool work cell is shown in Figure 22. Within the work cell space, 290 quasi-random axis-space points were generated to model the kinematic errors, and 50 quasi-random axis-space points were generated to validate the kinematic error model. Table 6 shows the minimum and maximum axes limits used in this study, and Figure 23 shows the distribution of model identification and model validation points with respect to machine table frame.

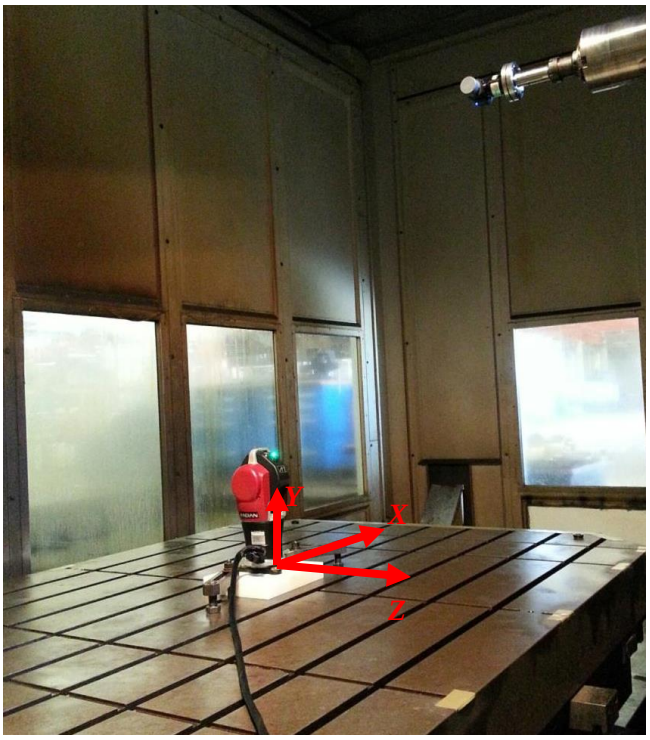


Figure 22: Machine tool work cell and table base frame

Table 6: Minimum and maximum commands used for modeling and validation.

Axis	Minimum Command	Maximum Command
B	0 deg	360 deg
X	-1250 mm	1250 mm
Z	900 mm	2200 mm
Y	350 mm	2500 mm
W	-800 mm	-200 mm

The point generation code developed creates pseudo-random points distributed throughout the joint space of the machine. The inputs available are the following:

Inputs: MeasPnts – Number of identification points to be generated; integer > 0

ID_file – File name to write identification points; string, should end in ".mpf"

ValPnts – Number of validation points to be generated; integer > 0

VAL_file – File name to write identification points; string, should end in ".mpf"

rng1 – X axis range to generate points over; vector with 2 elements

rng2 – Y axis range to generate points over; vector with 2 elements

rng3 – Z axis range to generate points over when B is near 0° or 180°; vector with 2 elements

rng3b – Z axis range to generate points over when B is near 90° or 270°; vector with 2 elements

rng4 – W axis range to generate points over; vector with 2 elements

rng5 – B axis range to generate points over; vector with 2 elements

longTool – estimated tool length when using long tool mount; scalar > 0

shortTool – estimated tool length when using short tool mount; scalar > 0

speed – maximum speed for each axis; in the order [X,Y,Z,W,B]

Outputs: Identification set and validation set data files

To prevent the laser tracker cord from wrapping itself around the tracker through continuous rotation of the machine B axis, modifications were made to the point generation such that rather than ordering the commands randomly and then using a nearest neighbor sort directly, the points are grouped into four quadrants of the B axis. Additional movement points are added between execution of each quadrant to rotate the table into the next quadrant in a controlled B axis direction. The B axis should always be rotated to near 0 degrees before running the program, which then rotates the table through approximately 270 degrees to the farthest quadrant before working back to the starting position. The identification and validation points used for the PAMA at the Caterpillar Tech Center are shown in Figure 23.

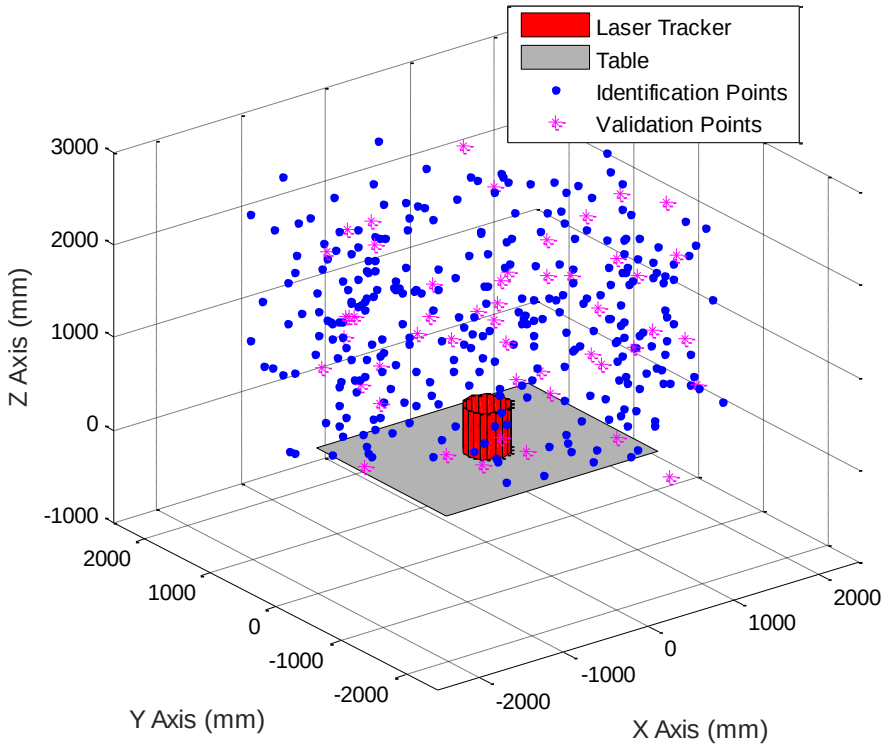


Figure 23: Display of Identification and Validation Point Distribution

After setup, the generated points are then measured with the laser tracker. The machine moves to each of the commanded points and the laser tracker measures the actual location of the active target. The difference between the desired location of the machine commanded and the actual location moved to is the residual error. Figure 24 illustrates the three types of residual error for each point. The nominal residual is that from the uncompensated actual measurement of the machine. An error model is then generated that attempts to fit a model to the measured residuals. The modeled residual is the predicted remaining error from each point's measurement if the model error at that point is removed via compensation. The Compensation residual is the residual of the actual measurements made on the machine after applying compensation tables.

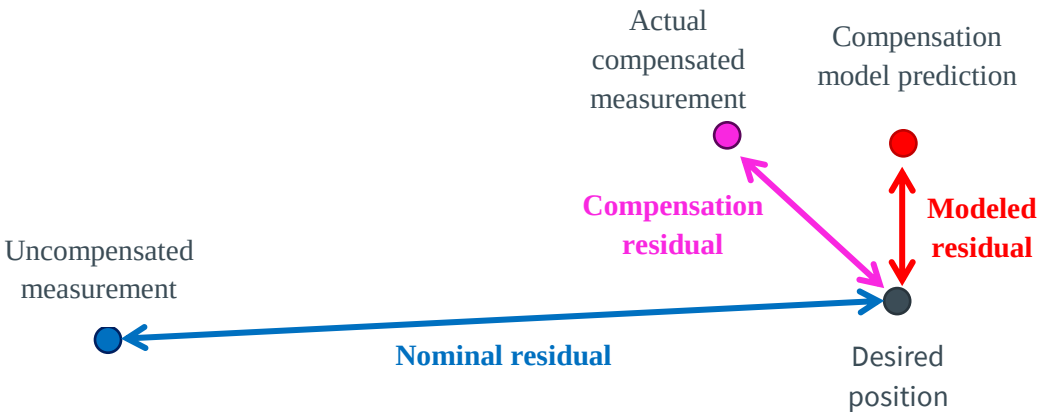


Figure 24: Illustration of Residuals

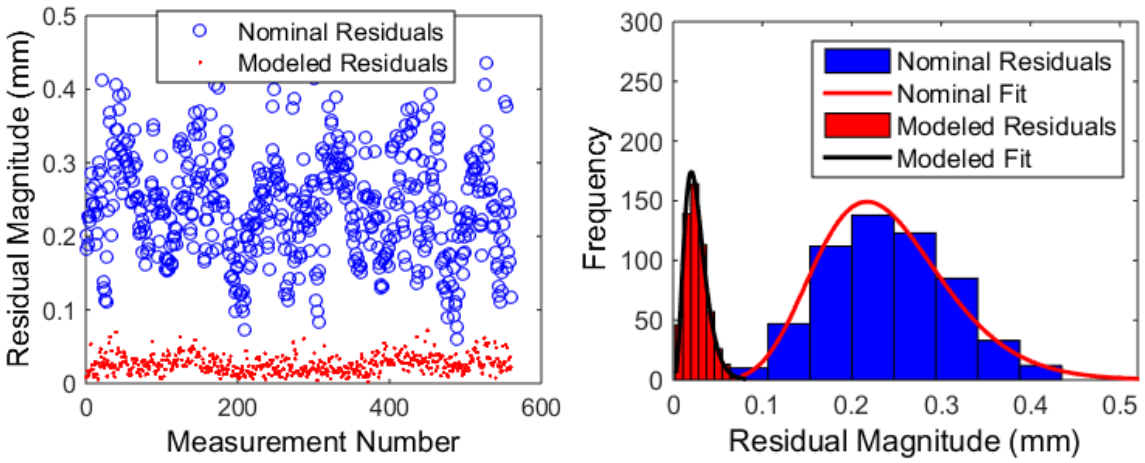


Figure 25: Sample Modeling Results

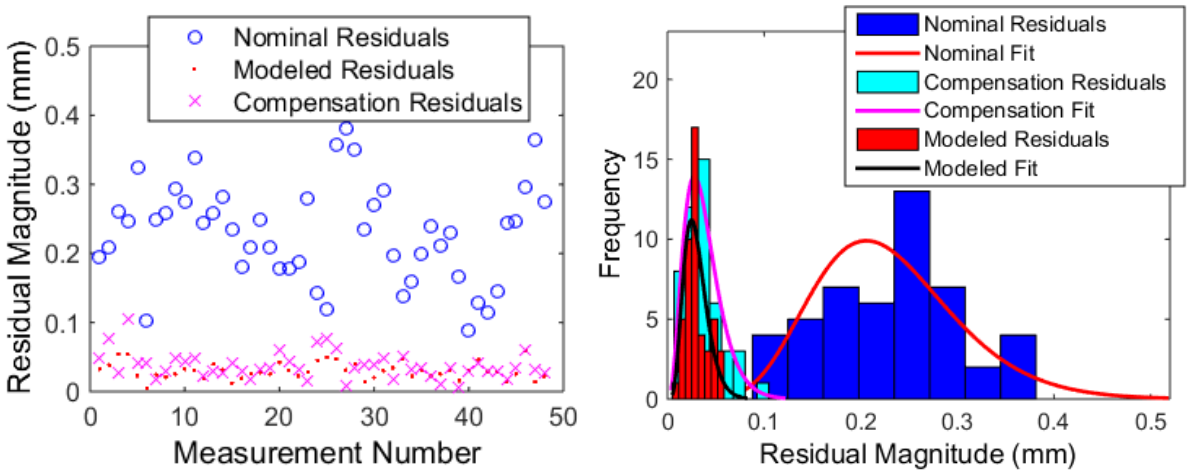


Figure 26: Sample Modeling Result with Compensation Validation Measurements

Compensation Functions

Compensation tables were generated from the model and uploaded to the machine. The validation set was re-measured with compensation enabled to test the validity of the compensation. The 25 error functions generated by the modeling algorithm are shown below in Figure 27. These functions are discretized in order to generate the compensation tables that are loaded onto the PAMA controller.

From Figure 27, there are three major features of the error functions to consider. First, these functions show that the compensation on the B axis is small with the largest error being 0.1 millidegrees. Additionally, the compensation functions change between the three compensation events with ZY, WY, and YX being the most similar between all three compensation events. Finally, the compensation on Z and W are nearly identical. This is most likely due to these axes being redundant, so the modeling algorithm divides the error evenly between these two axes.

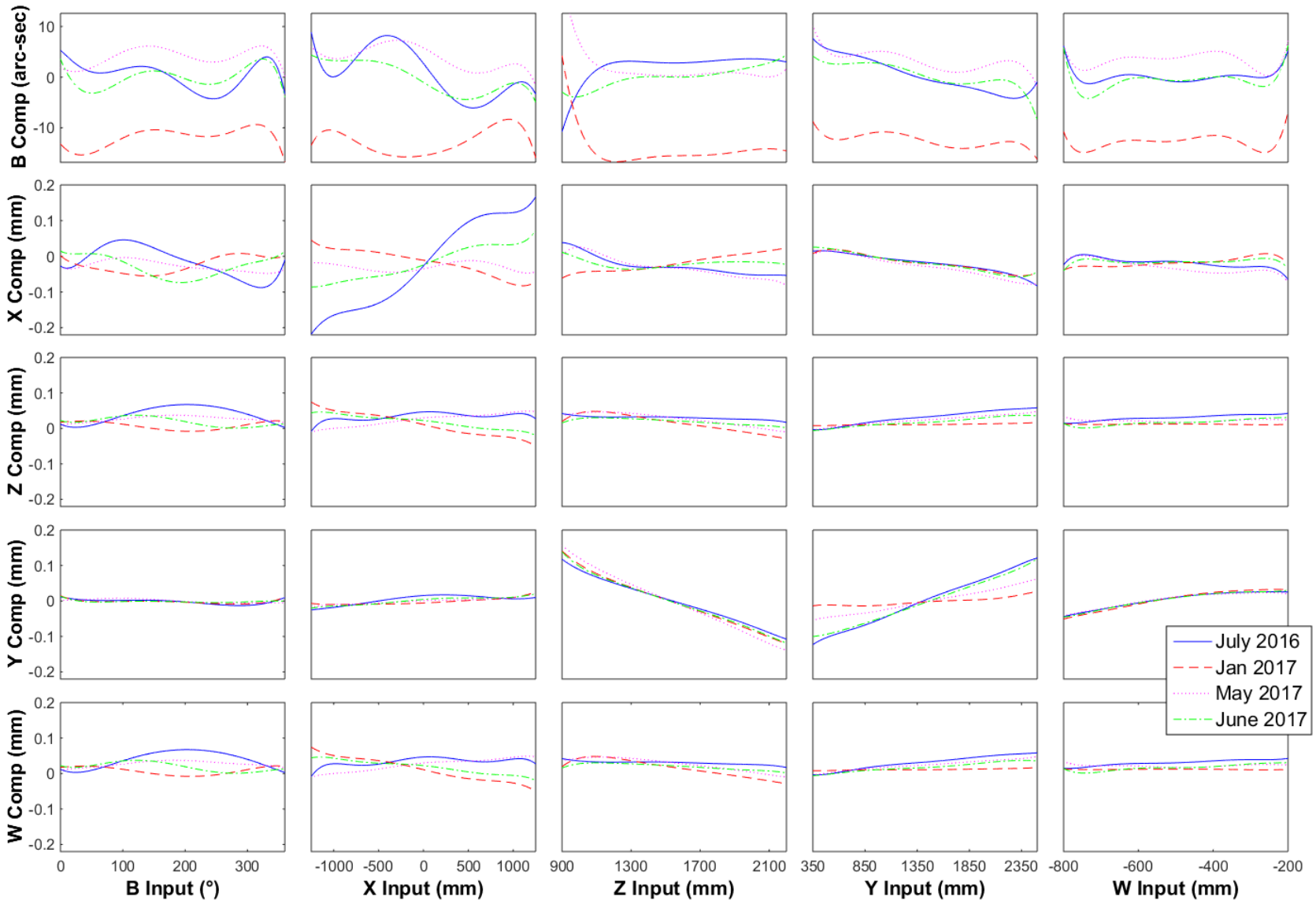


Figure 27: Identified PAMA error model functions.

Compensation Implementation Errors Corrected

The compensation tables generated during the previous Module 1 Final Test attempt in May 2017 did not result in the expected performance shown by the model. During the May event, it was determined that the compensation was not being applied based on either the Machine Coordinate System (MCS), or Work Coordinate System (WCS), but rather on an internal 'Position Actual Value Measurement System' (PAVMS) which had offsets in the Z and Y axes from the values used for the MCS. Additionally, the PAVMS for the B axis was found to extend indefinitely rather than always yielding a value between 0-360° as the MCS does. After identifying and correcting for these issues on May 5, the compensation reported on the machine was verified for all tables to be as expected, except for the tables which output compensation for the B axis. The compensation expected in the B axis was extremely small in the generated tables – barely above the minimal precision reported by the machine – so there was not enough precision to determine the root issue causing the error without loading a new set of tables with exaggerated compensation in the B axis. Implementation of the tables and off-site post event analysis revealed that the applied compensation appeared to be in the opposite direction of the reported compensation. Figure 28 shows the measured difference in the compensated versus the uncompensated validation set from the May 2017 event, as well as the expected compensation in the Y axis based on the tables.

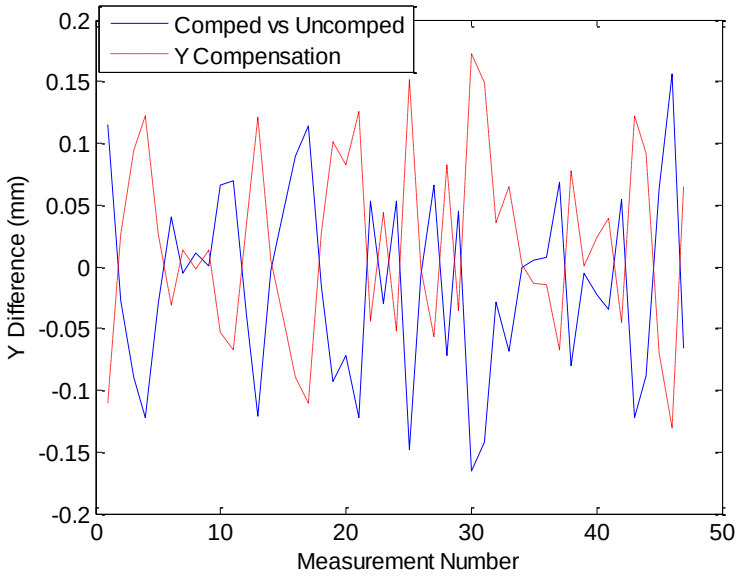


Figure 28: Difference between compensated and uncompensated validation sets compared with y compensation values.

During the June 2017 event the supposition that the applied compensation on the Tech Center PAMA machine was in the opposite direction of the compensation reported by the controller was confirmed. That is, the compensation value reported by the controller is from the comped value to the uncomped rather than the amount that was added to the uncomped position. To do this, the cross table that compensates the Y axis motion based on the Z axis position was uploaded to the machine tool controller, and six measurement location from the machine's workspace were

measured with the laser tracker each with the table enabled as well as with the table disabled. The difference between the compensated and uncompensated measurements of these six points matched the predicted compensation amount in Y within the repeatability of the machine indicating that the table was performing as expected. Once this compensation table had been verified, tables were uploaded for all axes except for the compensation onto the B axis. With all 20 of these tables, the same six points were measured with the laser tracker. As with before the measured compensation amount (this time in X, Y, and Z) was calculated and compared to the predicted compensation amount. These values again lined up within machine repeatability, which showed that the full set of tables performed as expected. After the other axes were verified, B axis tables were uploaded with compensation large enough to see on the controller. The controller recorded compensation value for each of these tables at the same six locations in the workspace were compared to the predicted compensation and lined up to the controller resolution. After the controller values for the B axis were verified, a setting in the controller was discovered to bound the B axis between 0 and 360 degrees, which resolves the issue of B axis inputs leaving the compensation table's range. Through this verification, it is possible to apply compensation to all axes on the PAMA machine.

4.1.3.6 B5 Comparison

To compare the quality of the measurements captured with the laser tracker for use in generating the VEC error models with traditional error measurement using B5 standard testing equipment, a measurement test of a body diagonal was taken with B5 equipment and then repeated using the laser tracker. Figure 29 shows the comparison between the two measurements. The left image shows the linear error measured in the body diagonal both with the compensation generated in a previous test turned on and off. It should be noted that the compensation had been generated under significantly different thermal conditions and was not designed to compensate for thermal error. The laser tracker measurements and the B5 measurements agree closely in both cases, with the histogram on the right showing that the maximum difference between the error calculated by each method was less than 15 microns.

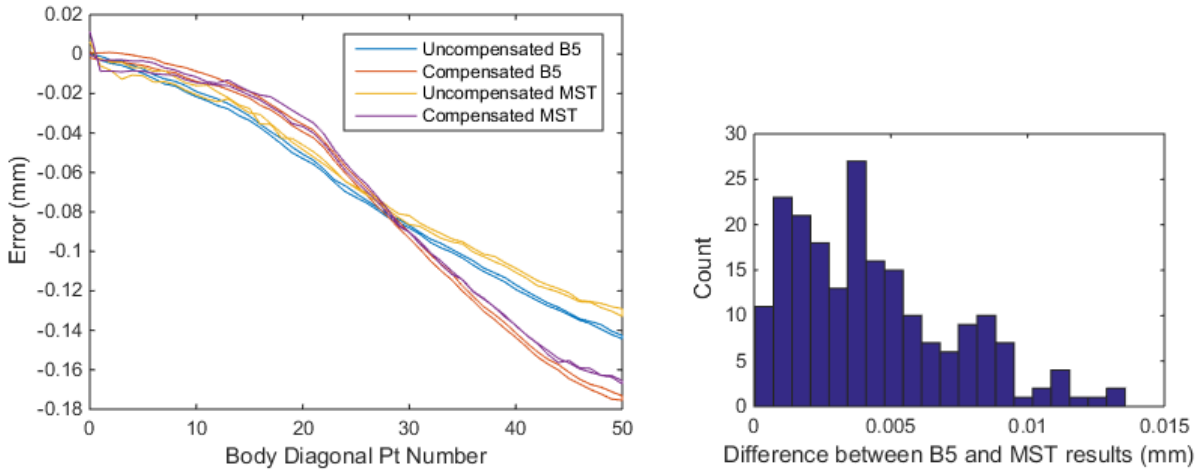


Figure 29: Comparison of laser tracker to B5 body diagonal measurement

4.1.4 Users & Use Cases

The VEC measurement and model generation was tested multiple times on the PAMA machine at the Caterpillar Tech Center with measurement events in July 2016, February 2017 and May 2017. Compensation was not implemented during these events. A full test of both measurement, error modeling, and validation of compensation was performed in June 2017. A summary of the results for the final event including validation of the compensation is given in Table 7. The time required for each portion of the process for this event is given in Table 8

Table 7: VEC measurement results

			99% Gamma		Mean	
Date	**Data Set**	**Type**	**Residual (mm)**	**% decrease**	**Residual (mm)**	**% decrease**
June 2017	Identification	Nominal	0.442	N/A	0.239	N/A
		Model	0.064	85.6%	0.026	89.1%
	Validation	Nominal	0.437	N/A	0.230	N/A
		Model	0.066	84.9%	0.030	86.9%
		Comp.	0.095	78.2%	0.038	83.7%

			99% Gamma		Mean	
Date	**Data Set**	**Type**	**Residual (mm)**	**% decrease**	**Residual (mm)**	**% decrease**
May 2017	Identification	Nominal	0.458	N/A	0.305	N/A
		Model	0.074	83.9%	0.030	90.0%
	Validation	Nominal	0.432	N/A	0.313	N/A
		Model	0.065	85.1%	0.030	90.4%
Jan 2017	Identification	Nominal	0.335	N/A	0.175	N/A
		Model	0.077	77.1%	0.029	83.7%
July 2016	Identification	Nominal	0.587	N/A	0.385	N/A
		Model	0.124	78.8%	0.053	86.2%
	Validation	Nominal	0.487	N/A	0.302	N/A
		Model	0.185	62.0%	0.095	68.5%

Table 8: VEC Process Timing

Activity	Start time	Approximate Duration
Warm up / Tool center check	7:19 AM	1 hr 5 min
Base frame and repeatability	8:24 AM	36 min
Machine down	9:00 AM	2 hr 50 min
Re-warm up	11:50 AM	50 min

Short tool ID set	12:40 PM	1 hr 20 min
Tool change / centering	2:00 PM	15 min
Warm up	2:15 PM	20 min
Long tool ID set	2:35 PM	1 hr 20 min
Validation (uncompensated)	3:55 PM	20 min
Total measurement time		**5 hr 16 min**
Compensation generation	4:15 PM	8 min
Compensation upload	4:23 PM	20 min
Warm up	4:43 PM	20 min
Validation (compensated)	5:03 PM	18 min
Total compensation time		**1 hr 6 min**
Total Work Time		**6 hrs, 25 min**

Error! Reference source not found. summarizes the Machine Tool Error Compensation success c riteria, the measured values and the resulting evaluation.

Table 9 VEC Success Criteria Summary.

Metric of machine error compensation	Baseline	Goal	Results	Validation Method
Time required for measurement	16 hours	Less than 6 hours	Less than 5.5 hours	Validated on 4-axis machine (4mx2.2mx2.5m)
Comp/uncomp mean residual improvement	-	More than 80%	85%	Validated on 4-axis machine (4mx2.2mx2.5m)
Difference from B5 test	-	Less than 20%	Less than 10%	Diagonal error comparison
Training time	24 hours +	Less than 16 hours	Less than 16 hours	Reviewed with technician

4.1.5 Software and System Requirements

The methodology was developed with the API Radian laser tracker and API Active Target as the measurement hardware and Spatial Analyzer as the metrology software used for collecting the measurement data from the laser tracker. The methodology could be easily adapted to any metrology hardware and software capable of recording with similar or better accuracy the absolute position of a target placed in the machine spindle as the machine moves throughout its workspace. A PC for acquiring measurements, calculating error models and generating compensation tables is needed, with a license for Spatial Analyzer (or equivalent metrology

software) and MATLAB. The standard MATLAB toolboxes are all that is required, with the custom functions which were written given in the following list:

4.1.5.1 List of Functions

Cheby.m

Inputs: C – coefficients of the Chebychev Polynomial; vector (row or column)

qbar – normalized joint angle; scalar between -1 and 1

Outputs: Val – value of the Chebychev Polynomial; scalar

Require Functions: N/A

Function Description: This function is used to calculate Chebychev Polynomial values which are used as the basis function for error modeling. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: N/A

CheckComp.m

Inputs: Robot – stores all machine dependent values; structure

pts – list of joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

tables – set of compensation tables; matrix: each column is a table, each row is a point in

that table. Table order is B→B, X→B, Z→B, Y→B, W→B, B→X, X→X,

Z→X, Y→X, W→X, B→Z, X→Z, Z→Z, Y→Z, W→Z, B→Y, X→Y, Z→Y,

Y→Y, W→Y, B→W, X→W, Z→W, Y→W, W→W

Outputs: Comp – total compensation on select points; matrix: each row is the compensation on

the corresponding set of joint commands in pts in the order [X,Y,Z,W,B]

Require Functions: N/A

Function Description: This function calculates the predicted compensation from the 840D controller at the given joint location with the inputted tables active. This code will throw an error if the inputted joint command is less than the minimum joint value in the Robot structure. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: For other, 5 axis machine no change necessary but be very careful about the order of joints, joint commands, and tables. For other machine, add or subtract the necessary number of axes. Again be very careful of the ordering.

CheckPoints.m

Inputs: MeasPnts – Number of identification points to be generated; integer > 0

ID_file – File name to write identification points; string, should end in ".mpf"

ValPnts – Number of validation points to be generated; integer > 0

VAL_file – File name to write identification points; string, should end in ".mpf"

rng1 – X axis range to generate points over; vector with 2 elements

rng2 – Y axis range to generate points over; vector with 2 elements

rng3 – Z axis range to generate points over when B is near 0° or 180°; vector with 2 elements

rng3b – Z axis range to generate points over when B is near 90° or 270°; vector with 2 elements

rng4 – W axis range to generate points over; vector with 2 elements

rng5 – B axis range to generate points over; vector with 2 elements

longTool – estimated tool length when using long tool mount; scalar > 0

shortTool – estimated tool length when using short tool mount; scalar > 0

speed – maximum speed for each axis; in the order [X,Y,Z,W,B]

Outputs: Identification set and validation set data files

Require Functions: *ForKinPAMA.m*, *nearNeighbor.m*, *niederreiter2_dataset.m*, *writegcode.m*

Function Description: This code generates two sets of quasi-random points, does a cursory collision check, sorts them, and writes the points to a file. It also outputs (to the command window) an approximate number of minutes required to measure each of these sets. This is written as a script. All data entry must be done in the code itself in the section titled "SETABLE VALUES".

Changes Necessary for Other Machines: The forward kinematics call and various static parameters based on the PAMA machine used in collision detection would need to be changed as well as the two different Z ranges used based on the B axis orientation.

ForKinListPama.m

Inputs: cmm – list of joint commands; matrix: each row is in the order [X,Y,Z,W,B]

tool – tool length used for nominal position calculations; scalar > 0

Outputs: nomMeas – nominal kinematic positions; matrix each row in the nominal position of the corresponding row of cmm in the form of [X,Y,Z,1]

Require Functions: *ForKinPAMA.m*

Function Description: This function calculates the nominal kinematic position for a given set of joint commands. This differs from *ForKinPAMA.m* in that it calculates positions for a sequence of commands, includes the tool length, and only returns the linear position as opposed to the full transformation matrix.

Changes Necessary for Other Machines: Change the call for *ForKinPAMA.m* to a different nominal kinematic function for the new machine.

ForKinPama.m

Inputs: q – single set of joint commands; vector in the order [X,Y,Z,W,B]

Outputs: Fn – nominal kinematic transformation matrix excluding tool length; matrix, 4x4

Require Functions: N/A

Function Description: This function calculates the nominal kinematic transformation matrix of the PAMA machine for a given set of joint commands. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: Kinematics need to be changed to match new machine.

ModelInput.m

Inputs: Measurement.ID.Cmm – joint commands for the identification set; matrix: each row is a set of commands [X,Y,Z,W,B]

Measurement.ID.Meas – measurements for the identification set; 3D matrix: each page uses a different tool length, each row is a set of measurements

[X,Y,Z]

Measurement.Base.Trans – nominal transformation from measurement frame to machine base frame; matrix: 4x4 typically identity

Measurement.Val.Cmm – joint commands for the validation set; matrix: each row is a set of commands [X,Y,Z,W,B]

Measurement.Val.Meas – measurements for the validation set; matrix: each row is a set of measurements [X,Y,Z]

Measurement.ID.MeasStd – measurement standard deviation matrix; matrix: each column corresponds to the measurement standard deviations for the corresponding row in the identification commands in the form [X1;Y1;Z1;X2;Y2;Z2;…(repeated for each tool used)]

Robot.Joint(i).JointLimits – joint limits for the i^{th} joint; vector with 2 elements

Robot.Joint(i).CmmStd – command standard deviation for the i^{th} joint; scalar

Outputs: Robot – stores all machine dependent values; structure

Measurement – stores all measurement dependent values; structure

Require Functions: *ModelMinimization.m*

Function Description: This function is the data entry point for the VEC modeling process. Once all values are populated and correct, run this script to do the modeling. If you would like to model without using a validation set, set Measurement.Val.Cmm to Measurement.ID.Cmm and Measurement.Val.Meas to Measurement.ID.Meas(:,:,2) as leaving the validation fields blank can cause the code to terminate with error.

Changes Necessary for Other Machines: Change the axis information under “Robot Structure” to match the new machine.

ModelMinimization.m

Inputs: Robot – stores all machine dependent values; structure

Measurement – stores all measurement dependent values; structure

Outputs: Robot – stores all machine dependent values; structure

Measurement – stores all measurement dependent values; structure

Require Functions: *ModelResidual.m*

Function Description: This function is where most of the back end calculation for the model fitting process takes place. It identifies the error model parameters and calculates the model residuals. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: Edit "Extra Data" and "Preprocessing" sections to line up with the new machine configuration. If the number of joints changed, up_cmm (line 80), up_meas (line 81), low_cmm (line 85), and low_meas (line 86) would have to change their reference ranges accordingly as well as the references ranges of x used to calculate upper (line 97), lower (line 100), and F_resid (line 105).

ModelResidual.m

Inputs: Robot – stores all machine dependent values; structure

Measurement – stores all measurement dependent values; structure

cmm – set of joint commands; vector in the order [X,Y,Z,W,B]

meas – set of measurements at the given joint commands; vector in the form

[X1,Y1,Z1,X2,Y2,Z2,…(repeated for each tool used)]

Para – list of model parameters; vector with the first 6 elements corresponding to the base frame error, the last 3*(number of tools) corresponding to the tool length error, and the rest corresponding to joint error

Outputs: resid – residual error between the VEC model and the nominal position; vector in the form [X1;Y1;Z1;X2;Y2;Z2;…(repeated for each tool used)]

Require Functions: *Cheby.m*

Function Description: This function calculates the residual between the VEC model and the nominal position for each tool length used. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: Edit the "Forward Kinematics" section to line up with the new machine configuration.

nearNeighbor.m

Inputs: cmm – list of joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

Outputs: CMM – sorted list of joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

Require Functions: N/A

Function Description: This function does a nearest neighbor sorting of sets of joint commands in order to reduce the required measurement time. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: For machines with a different number of joints, the indices in line 11 need to be changed.

niederreiter2_dataset.m

Inputs: m – dimension of the space to generate points in (e.g. number of joints); string representing integer > 0 (e.g. '1')

n – number of points to generate; string representing integer > 0

skip – number of initial values to skip (acts as a seed); string representing integer > 0

Outputs: r – set of generated data points; matrix: mxn

Require Functions: N/A

Function Description: This function generates sets of quasi-random points which are used as identification or validation sets. The points that it outputs are between 0 and 1, so they must be scaled to the appropriate joint ranges. This code is formatted as a function and can be called from other functions.

Changes Necessary for Other Machines: N/A

REPEATABILITY_NEW.m

Inputs: RP – set of repeatability measurements; matrix: each row represents a measurement in

the form [X,Y,Z]

p – number of different measurement locations used; integer > 0

c – number of measurements taken at each location; integer > 0

Outputs: N/A

Require Functions: N/A

Function Description: This function calculates the repeatability of a set of measurements and generates a figure of the repeatability measurements. It is important that each measurement location in visited the same number of times and that the product of c and p is equal to the number of rows in RP. If this is not true, the code will give false results.

Changes Necessary for Other Machines: N/A

SeporateTables.m

Inputs: Robot – stores all machine dependent values; structure

Outputs: Table – list of generated tables; matrix: each column is a table, each row is a point in that table. Table order follows joint order in Robot.

Require Functions: N/A

Function Description: This function generates 25 different compensation table files (from each axes to each axis). These files are named TAB0.INI through TAB24.INI in accordance with the files set up by PAMA. Each table contains 1000 points, but this can be changed by changing the Num_Pts variable on line 3 (Note that all tables must have the same number of points in this code). While this code generates both INI files and a matrix output, the matrix is only necessary for troubleshooting (i.e. as an input to *CheckComp.m*). This code is formatted as a function and can be called from other functions, but it is currently not directly implemented in *ModelInput.m*.

Changes Necessary for Other Machines: This code currently generates 25 tables, so if the number of axes change, the number of loop iterations (in line 9) must also change. The file names (constructed on lines 10 through 14) may also need to change when implemented on other machines. On other machines, the variables AXES (line 6) and AX (line 7) should line up with the controller names for each axis (in the order used in Robot). This table format is specifically designed for the Siemens 840D controller and may not work on machines with other controllers.

SortID.m

Inputs: Mcmm – list of identification set joint commands; matrix: each row is a set of commands

[X,Y,Z,W,B]

ShortTool – tool length of the short tool; scalar >0

LongTool – tool length of the long tool; scalar >0

ID1 – set of identification measurements generated with the short tool; matrix: each row

is a set of measurements [X,Y,Z] (does not need to match order of Mcmm)

ID2 – set of identification measurements generated with the short tool; matrix: each row

is a set of measurements [X,Y,Z] (does not need to match order of Mcmm)

Outputs: CMM – sorted list of identification set joint commands; matrix: each row is a set of

commands [X,Y,Z,W,B]

Short – sorted set of identification measurements generated with the short tool; matrix:

each row is a set of measurement [X,Y,Z] corresponding to the order of CMM

Long – sorted set of identification measurements generated with the long tool; matrix:

each row is a set of measurement [X,Y,Z] corresponding to the order of CMM

Require Functions: *ForKinListPAMA.m*

Function Description: This function lines up the data between both identification measurement sets and the identification joint commands. This is necessary because many times in the measurement process extra measurements will be taken or measurement locations will be skipped. To line up the data, measurements are compared to the nominally commanded position with a tolerance of 5 (typically mm but will match whatever units are being used. If this tolerance is too broad or tight, it can be changed in lines 17 and 22. To use this code, the input variables names can be changed to match the data, but it is generally easier to name the raw data to match the inputs.

Changes Necessary for Other Machines: Change the calls for *ForKinListPAMA.m* to a different nominal kinematic function for the new machine.

SortVAL.m

Inputs: Vcmm – list of validation set joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

LongTool – tool length of the long tool; scalar >0

V1 – set of validation measurements generated with the long tool; matrix: each row is a set of measurements [X,Y,Z] (does not need to match order of Vcmm)

Outputs: VCMM – sorted list of validation set joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

vresid – set of nominal residuals for each validation point; vector: each row corresponds to the joint command in the same row of VCMM

Vdata – sorted set of validation measurements generated with the long tool; matrix: each row is a set of measurement [X,Y,Z] corresponding to the order of CMM

Require Functions: *ForKinListPAMA.m*

Function Description: Functions the same as *SortID.m* except it only takes in one set of measurements and also outputs a set of nominal residuals for that data set.

Changes Necessary for Other Machines: Change the call for *ForKinListPAMA.m* to a different nominal kinematic function for the new machine.

ValResid.m

Inputs: Vcomp – set of compensated validation measurements generated with the long tool; matrix: each row is a set of measurements [X,Y,Z] (Must be sorted to match Measurement.Val.Cmm)

Robot – stores all machine dependent values; structure

Measurement – stores all measurement dependent values; structure

Outputs: vresidc – set of residuals between compensated measurements and nominal positions; vector: each row corresponds to the joint command in the same row of Measurement.Val.Cmm

Require Functions: *ModelResidual.m*

Function Description: This function calculates the residual of the compensated measurements and generates plots with the nominal, modeled, and compensated validation set residuals.

Changes Necessary for Other Machines: N/A

writegcode.m

Inputs: q – list of joint commands; matrix: each row is a set of commands [X,Y,Z,W,B]

filename – name of .mpf file to write commands to; string ending with ".mpf"

dwell – Boolean vector to determine if the machine should stop at the corresponding point; vector 1 if the machine should stop at the corresponding row of q and 0 if the machine should not stop

Outputs: N/A

Require Functions: N/A

Function Description: This function generates a set of gcode to move through the specified sequence of joint command. At each joint location the machine can wait for 7 seconds (i.e. to take measurements) or move directly to the next location (i.e. for transition points).

Changes Necessary for Other Machines: Axis commands used in line 34 will need to be changed to match with the new machine's controller.

4.1.6 Features and Attributes

The system used for machine tool error measurement, modeling and compensation has the following attributes:

1. Measures absolute error rather than error relative to an arbitrary axis starting position.
2. Measures error simultaneously for all axes, throughout the workspace rather than each axis independently with other axes held constant. This results in an error model that is not overfit to a specific location in the workspace.
3. Detailed compensation tables of 1000 points each are used including tables compensating for both linear error within an axis and cross-axis error dependence between all axis combinations.
4. Methodology is easily adaptable to any laser tracker and active target system which allows the target to be mounted in the spindle of the machine and the absolute position

of the target relative to the tracker measured throughout the workspace of the machine with similar or better accuracy to that of the API Radian and API active target.
5. Pseudorandom point sets used for testing are clustered into quadrants and the B-axis motion is moved between quadrants in a designed direction to avoid cord-wrap problems.
6. Software was customized for a 5-axis BXYZW style kinematic structure, but the general methodology is applicable to any kinematic structure with the proper code modifications.

4.1.7 Modes of Operation

This technology can be used in new machine tool asset calibration as well as recalibration of the degraded machine tool assets. Laser tracker mount and calibration is need for the hardware preparation. Then, laser tracker measurement tracking performs with the support from operator. With the laser tracker measured data, software generates 25x1000 point compensation point and applied to the controller by manufacturing engineer or operator.

4.2 Thermal error measurement

Another main difficulty in machine tool calibration is the fact that quasi-static effects are the dominant contributors to volumetric errors observed in the workspace. As, a result, thermal and flexural deformations combine with manufacturing and assembly errors of machine elements and through complex kinematic transformations reveal themselves in the volumetric errors of a machine. Thus, the first problem to be addressed is one of modeling how the dimensional and assembly errors of a machines elements kinematically compose the volumetric errors. This report gives a systematic and mechanized process for composing error models. While the modeling approach is not new, extensions are provided to include rotary joints and modeling procedures are updated to be able to exploit the speed and versatility of modern metrology instrumentation, especially laser trackers.

4.2.1 System overview

Kinematic error models for multi-axis machine tools will typically involve several unknown (component error) parameters. A large number of volumetric error observations must be made to robustly estimate these errors. The ease of use and speed of laser trackers open up the possibility of making a large number of volumetric error measurements across a machine tool's workspace to estimate the unknown parameters its kinematic error model. The procedure for using laser tracker information to identify the unknown parameters (component errors) of the kinematic error model is developed, tested, and shown to be effective in characterizing the observed volumetric errors.

Thermal effects cause a machine tool's volumetric error to change during operation. Depending to factory environment, the severity of the work-cycle, and the thermal susceptibility of machine's design, these changes can be as large, if not larger, than the errors attributed to the

dimensional and assembly tolerances of the machine. Notwithstanding the speed and convenience of laser trackers, the sheer number of observations needed for repeatedly estimating kinematic model error parameters to track the thermal drift of a machine by periodically rebuilding the error model make it infeasible to use the same measurement and identification strategies used for machine calibration. First, this causes a severe reduction in the productive time of the machine tool when done at reasonable intervals to have a meaningful compensatory effect. Second, the transient nature of a machine's thermal state makes estimates of model parameters based on observations an over long interval of time questionable. Therefore, the concept of optimal error observers has been introduced. Given a fixed number of observations (much smaller than that needed for calibration), the optimal error observer, based on the structure of the kinematic error model, locates these observations in the machine's workspace so as to produce the best-conditioned/most informative relational matrix for the identification of the error model's unknown parameters. Limiting the number of observations needed for these estimates shortens the length of the non-productive machine time interval needed for measurements, making it feasible to periodically update the error model while the machine is in operation.

What are quasi-static machine tool errors?

About 70-80 percent of the inaccuracy of a machine tool is caused by quasi-static errors. As their name suggest, the quasi-static errors are slowly varying errors. Quasi-static error sources include assembly errors, flexural errors (due to self-weight of moving parts and the work piece), and thermal deformations (due to heat generation at the spindle, drives, guideways and cutting tools as well as ambient temperature variations, all of which gradually generate the geometric inaccuracies in the underlying kinematic structure of the machine). Compared with dynamic errors (e.g., servo-tracking errors; dynamic response to cutting forces), quasi-static errors vary slowly during the operation of the machine. Due to being constrained, small thermal changes cause structural members of the machine to undergo deformation that, in turn, are magnified by the Abbe effect across other structural members. Therefore, depending on the mode of operation and the level of control of the factory environment, thermal (drift) errors can become the dominant component of quasi-static errors, especially for larger machines with variable operation cycles. For example, on a shop floor with controlled temperature and the machining operating continuously on a repeating work cycle, a thermal steady state is reached and thermal drift of the machine is minimal. However, for a machine operating in a flexible (small-batch/one-off) production setting, without shop floor temperature controls and large diurnal temperature swings, the quasi-static errors of the machine may be very large.

4.2.2 System Architecture

The flowchart in Figure 30 shows methodology developed along with the software modules (described later) developed to support its use. The left column of the flow-chart is concerned with the development of the kinematic error model. The right column of the chart defines the

steps used to construct the error observer for updating the error model during operation. The two steps at the bottom of the chart are the parameter estimation process.

Kinematic error modelling, for a five-axis machine

Model construction is shown below.

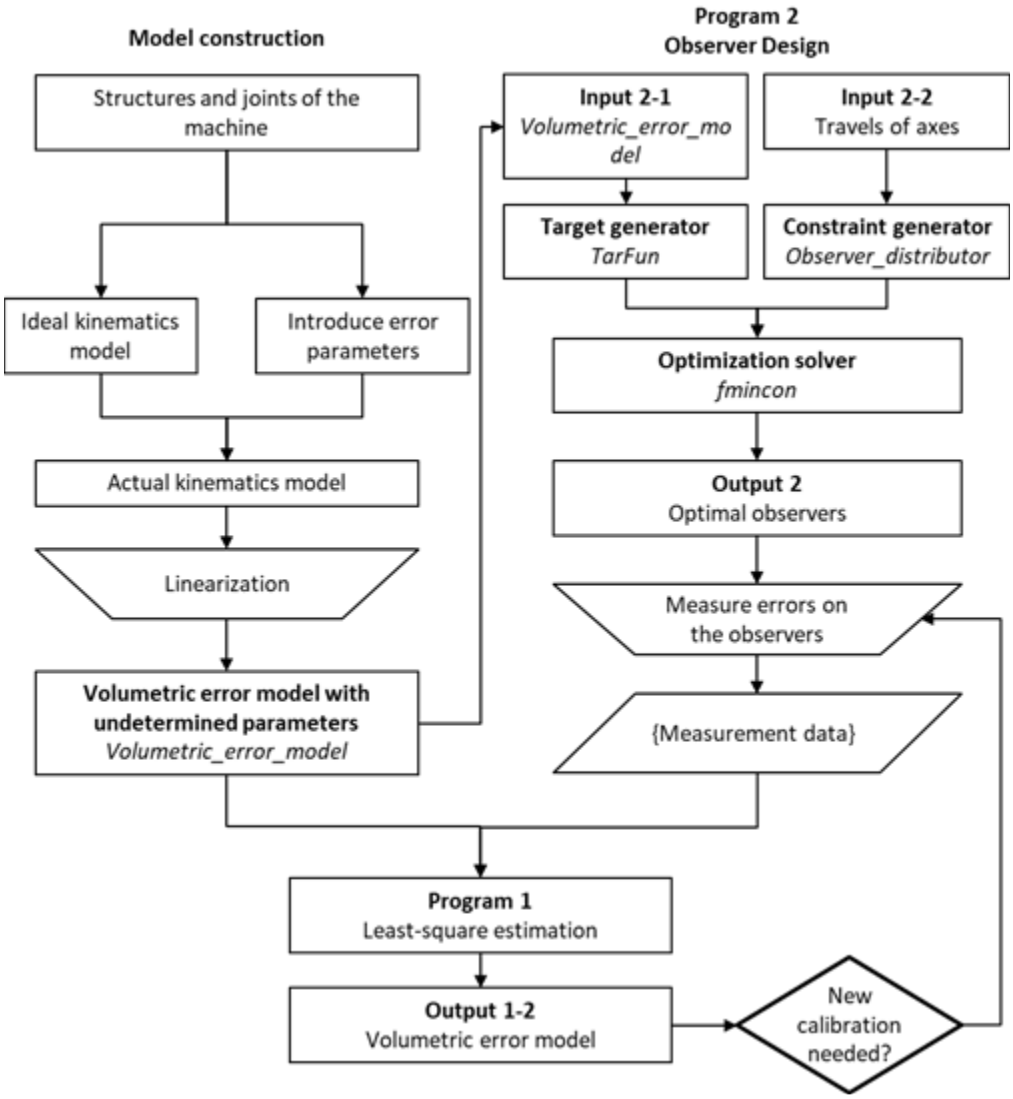


Figure 30 **Flowchart to operate the system**

This section describes the modeling approach, which is shown in the left hand side in Figure 30. As shown in Figure 31, the ideal kinematic of the machine from the table to the spindle can be expressed by the series of homogeneous transformation matrices (HTMs). This series consists of alternating joint and shape transformations. The machine has four prismatic joints, X, Y, Z and W axis with travels of 4, 2.5, 2.2 and 0.8 m respectively and a rotary joint, B axis that allows the table

to rotate about the Y direction by 360 degrees. The Z and W axes are redundant axes. The error model is obtained by introducing perturbations to the ideal kinematics of all the elements of the machine's kinematic chain, from the table to the spindle. Expressed by the series of homogeneous transformation matrices (HTMs),

$$\begin{bmatrix}\vec{e}\\1\end{bmatrix} = T_0(R-H)\begin{bmatrix}\vec{r}_t\\1\end{bmatrix} = T_0\Delta H\begin{bmatrix}\vec{r}_t\\1\end{bmatrix}, \qquad (37)$$

where $\vec{e} = \begin{bmatrix}e_x\ e_y\ e_z\end{bmatrix}^T$ is the error vector, $H = \Phi_B T_1 \Phi_x T_2 \Phi_z T_3 \Phi_y T_4 \Phi_w T_5$ and $R = (\Phi_B + \Delta\Phi_B)(T_1 + \Delta T_1) \dots (T_5 + \Delta T_5)$ are the ideal and perturbed forward kinematics, $\vec{r}_t$ is the position of the target in the spindle frame, T_0 is a fitted HTM that depicts the displacements between the nominal and actual measurement frame and ΔH is the error kinematics obtained by eliminating the higher order terms in the kinematic chain.

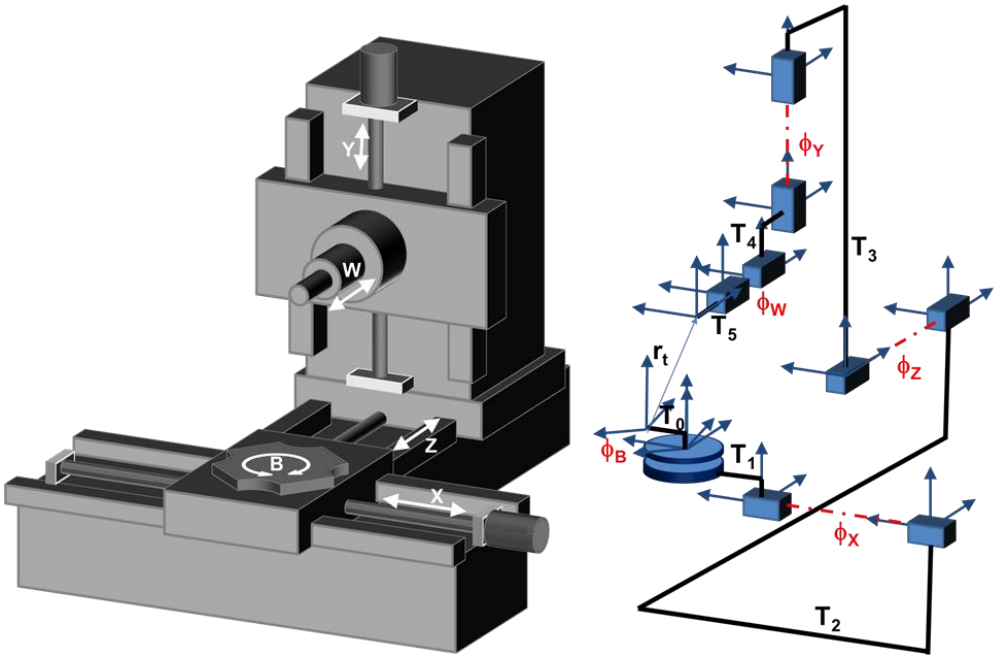


Figure 31 **Schematic of a 5-axis machine along with kinematic model showing the shape and joint transformation**

There are three types of transformation for a machine tool. The constant (not position dependent) components of errors are introduced into the shape transformations while the position dependent components are introduced into the joint transformations. The error parameters in three types of transformations are introduced and explained as follow.

1. Shape transformations, $T_i(\alpha_i, \beta_i, \gamma_i, \Delta x_i, \Delta y_i, \Delta z_i), i = 1\sim5$

For an inaccurate shape transformation as shown in Figure 32, Δx_i, Δy_i and Δz_i depict the dimensional (translation) errors while α_i, β_i and γ_i capture the deflection (angular) errors (roll, pitch and yaw).

$$T_i' = T_i + \Delta T_i = \begin{bmatrix} 1 & -\alpha_i & \beta_i & x_i + \Delta x_i \\ \alpha_i & 1 & -\gamma_i & y_i + \Delta y_i \\ -\beta_i & \gamma_i & 1 & z_i + \Delta z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (38)$$

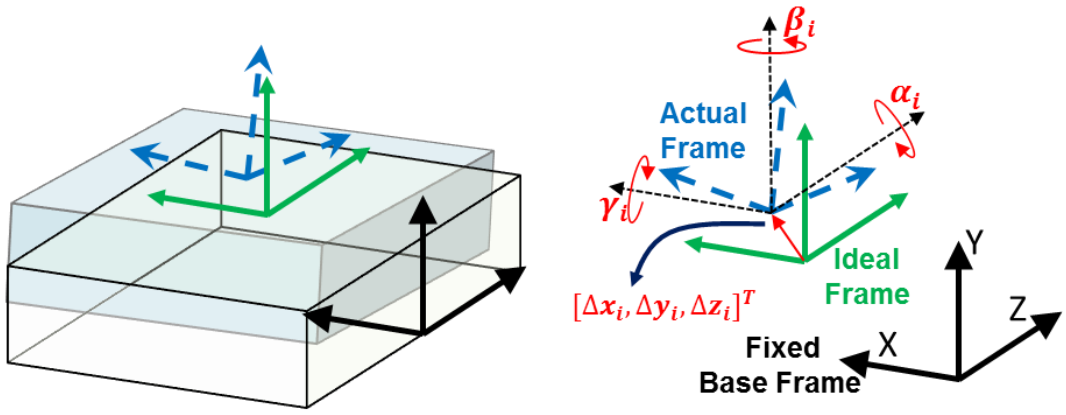


Figure 32 **Ideal and actual shape transformations**

2. Rotary joint transformation, B-axis, $\Phi_B\ (\alpha, \beta, \gamma, d_x, d_y, d_z)$

Figure 33 shows a rotary joint, which is commanded to an angular position B (rotation about Y-axis) may have positioning error, $\beta B'$. Further, the rotational errors may introduce tilts of $\alpha B'$ and $\gamma B'$and the entire moving table may shift due to the accumulation of translation errors B'^{d_x}, B'^{d_y}, B'^{d_z}, where $B' = \sin\left(\frac{B}{2}\right)$. All translational and rotational errors are modelled by Fourier sine series of B. For simplicity, only the first term in the Fourier series is taken.

$$\Phi_B' = \Phi_B + \Delta\Phi_B = \begin{bmatrix} cos(B + \beta(B)) & -\alpha(B) & sin(B + \beta(B)) & d_x(B) \\ \alpha(B) & 1 & -\gamma(B) & d_y(B) \\ -sin(B + \beta(B)) & \gamma(B) & cos(B + \beta(B)) & d_z(B) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (39)$$

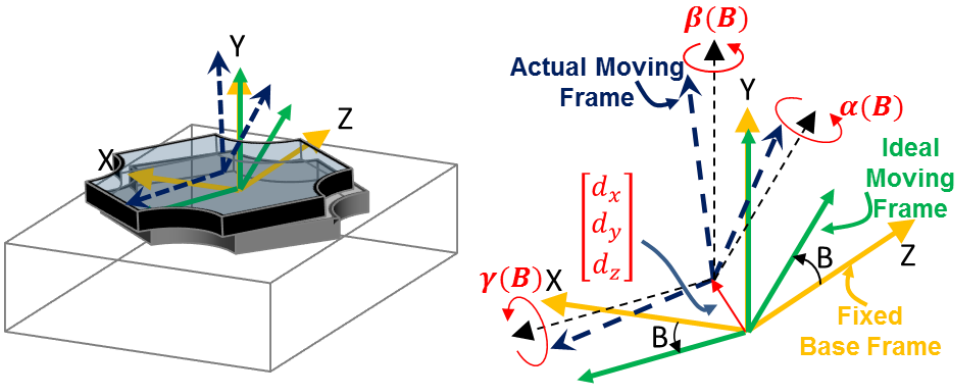


Figure 33 **Ideal and actual rotary joint transformations**

3. Prismatic joint transformations, X-axis, $\Phi_x(\delta x, \frac{d\alpha}{dx}, \frac{d\beta}{dx}, \frac{d\gamma}{dx})$, Y-axis, $\Phi_y(\delta y, \frac{d\alpha}{dy}, \frac{d\beta}{dy}, \frac{d\gamma}{dy})$, Z-axis, $\Phi_z(\delta z, \frac{d\alpha}{dz}, \frac{d\beta}{dz}, \frac{d\gamma}{dz})$ and W-axis, $\Phi_w(\delta w, \frac{d\alpha}{dw}, \frac{d\beta}{dw}, \frac{d\gamma}{dw})$

Similarly, the actual prismatic joint has the error in positioning along the joint and angular errors, which are all modelled as linear function of the joint command. For example, the positioning error is the x times δx and the angular errors are x times $\frac{d\alpha}{dx}, \frac{d\beta}{dx}$ and $\frac{d\gamma}{dx}$, respectively. The linear variation of angular errors with displacement along the axis necessitates the addition of squared terms to the straightness error.

$$\Phi_x' = \Phi_x + \Delta\Phi_x = \begin{bmatrix} 1 & -x\frac{d\alpha}{dx} & x\frac{d\beta}{dx} & x + x\delta x \\ x\frac{d\alpha}{dx} & 1 & -x\frac{d\gamma}{dx} & \frac{x^2}{2}\frac{d\alpha}{dx} \\ -x\frac{d\beta}{dx} & x\frac{d\gamma}{dx} & 1 & -\frac{x^2}{2}\frac{d\beta}{dx} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (40)$$

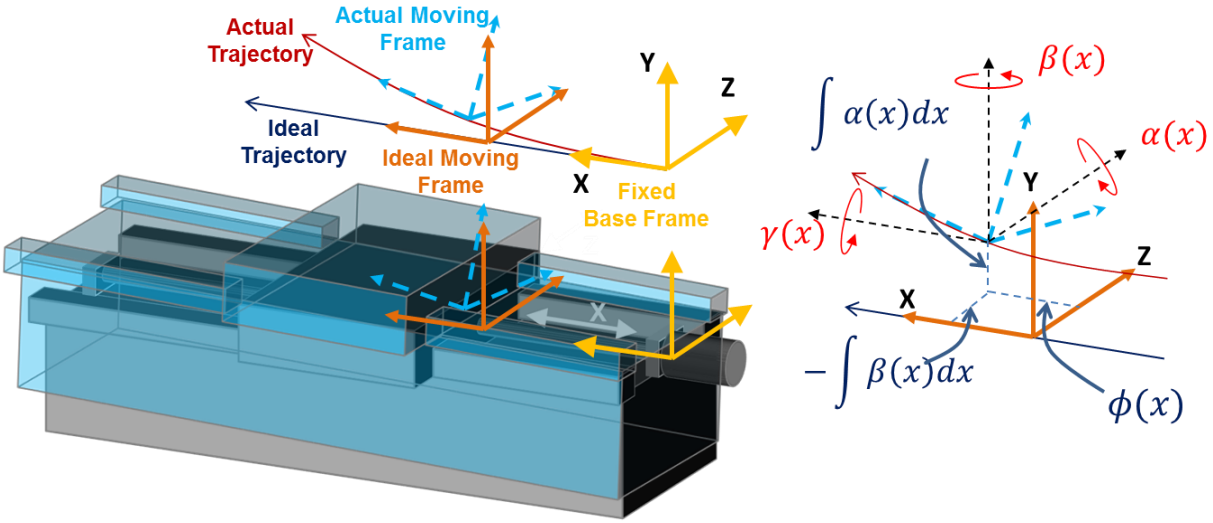


Figure 34 **Ideal and actual prismatic joint transformations**

The 52 introduced error sources/parameters are assumed to be small. Therefore, an error model in the set of parameters and can be expressed by the difference between the ideal and actual forward kinematics. By ignoring the higher order terms and separating the contribution of each error parameter on the error vector, the linear error model can be written as $\vec{e}_i = \mathcal{M}_i'\vec{p}'$ where $\vec{e}_i = [e_{x,i}\ e_{y,i}\ e_{z,i}]^T$ contains the components of the errors observed at that point, $\vec{p}'$ contains all 52 error parameters and $\mathcal{M}'$ is a matrix with three rows and 52 columns, where each term being a function of known machine constants and commanded positions.

As would be expected, the influence of some parameters on the observed volumetric error components will be inseparable from, or linearly dependent on, each other when only changes

in the commanded position of the tool are introduced. For example, Δx_i and β_i's contributions to the volumetric error components at a point are inseparable, irrespective of its location in the machine's workspace. They must therefore be identified as a group. Further, other parameters such as α_4,α_5 and $\frac{d\alpha}{dw}$ have no influence on the volumetric error when the tool reference point lies along the axis of the spindle. In all, our analysis has found that with such groupings and eliminations, there are 32 identifiable error parameters. After all these redundant parameters are eliminated or grouped, we have

$$\vec{e}_i = \mathcal{M}_i \vec{p} \qquad (41)$$

where $\mathcal{M}_i \in \mathbb{R}^{3\times 32}$ is a sub-matrix of $\mathcal{M}_i'$,
$\vec{p} = \left[\alpha_1, \alpha_2 + \alpha_3, \beta_1, \beta_2, , \beta_3 + \beta_4, \gamma_1 + \gamma_2, \gamma_4, -\beta_5 r_t + \sum_{i=1}^{5} \Delta x_i\,, -\gamma_5 r_t + \sum_{i=1}^{5} \Delta y_i\,, \sum_{i=1}^{5} \Delta z_i\,, \frac{d\alpha}{dx}, \frac{d\beta}{dx}, \frac{d\gamma}{dx}, \delta x, \frac{d\alpha}{dy}, \frac{d\beta}{dy}, \frac{d\gamma}{dy}, \delta y, \frac{d\alpha}{dz}, \frac{d\beta}{dz}, \frac{d\gamma}{dz}, \delta z, \frac{d\beta}{dw}, \frac{d\gamma}{dw}, \delta w, \alpha, \beta, \gamma, d_x, d_y, d_z\right]^T \in \mathbb{R}^{32}$ and r_t is the tool length.

4.2.3 System development

4.2.3.1 Estimation of the error model parameters

The model, developed in the previous section, assembles all the error sources in the kinematic chain to obtain their influence on the volumetric error components of the machine. There are a total of 52 error sources or parameters (five shape transformations, each with six error parameters, four joint transformations for linear axes, each with four error parameters, and one for a rotary axis with six parameters) that are composed into an expression for the volumetric error components observed in the machines workspace. To use this model for compensating the volumetric errors, it is necessary to obtain values for these parameters.

Estimation of the parameters in the error model is done by observing the volumetric errors of the machine at different points in its workspace with a laser tracker, as shown in Figure 35. However, to do so, the frame in which the laser tracker makes measurements, T_0 must be first estimated before the parameters of the error model can be obtained. *This is done* in the following two steps.

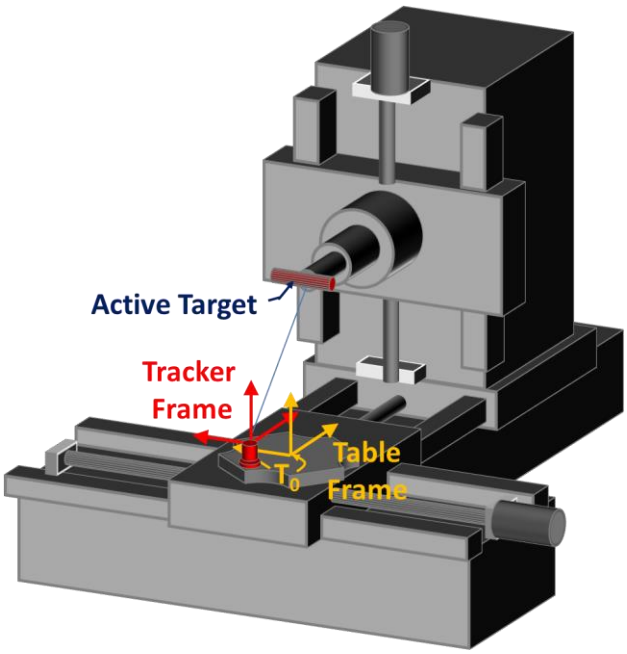


Figure 35 Schematic depiction of measurement of volumetric error components of the machine using a laser tracker. The relationship between the measurement frame and the table frame (from the kinematic model of the machine) is captured by the homogeneous transformation matrix T_0.

1. Find T_0 , the best-fit measurement frame

Assume the machine to be ideal and identify the best values for T_0 to minimize the discrepancy between the laser tracker observations of position and the commanded position. Since T_0 is a rigid transformation, this step accounts any location and alignment errors between the machine and the laser tracker. In addition, this step will also reduce the effects of any error sources that produce a rigid displacement of the entire machines workspace. The residual errors that result from this process (of aligning measuring frame with the machines coordinates) are referred to as the nominal errors of the machine.

To identify T_0, assume ideal kinematics for the machine defined in equation (42),

$$\vec{r}_{t0} = T_0\Phi_B T_1\Phi_x T_2\Phi_z T_3\Phi_y T_4\Phi_w T_5 = T_0 H\vec{r}_t, \qquad (42)$$

where T_0 has rigid body translations and small angle rotations as parameters to be identified.

$$T_0 = \begin{bmatrix} 1 & -\alpha_0 & \beta_0 & x_0 \\ \alpha_0 & 1 & -\gamma_0 & y_0 \\ -\beta_0 & \gamma_0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (43)$$

Further, $\vec{r}_t$ is the position of the target in the spindle frame and $\vec{r}_{t0}$ is its image in the measurement frame. For different joint commands (or measurement points), the kinematic transmission of the machine, H will vary. For the i[th] measurement point, the error vector, $\vec{e}_i$ between the forward kinematic transmission and the measurement recorded by the laser tracker can be expressed as

$$\vec{e}_i = T_0 H_i \vec{r}_t - \vec{q}_i, \tag{44}$$

where $\vec{q}_i$ is the measurement recorded by the tracker.

The best-fit homogeneous transformation, T_0 to the measurement frame can be obtained by minimizing the sum-of-squares of the discrepancy between the ideal machine's commanded positions and the measurements made by the tracker,

$$\min_{T_0} \sum_{i=1}^{n} \|T_0 H_i \vec{r}_t - \vec{q}_i\|^2 \tag{45}$$

2. *Identify the parameters of the error model from the nominal errors observed in the machine's workspace*

The error sources in the kinematic chain of the machine cause the workspace of the machine to dilate/contract, shear and bend. These effects are encoded errors measured in the point-cloud of error measurements made by the laser tracker. In this step, least-square is used to identify the parameters. Consider a typical linear identification/design of experiments problem with n design points where we have a random process:

$$\vec{e} = M(\vec{j}_1, \ldots \vec{j}_n)\vec{p} + \vec{N} \tag{46}$$

where $\vec{e} \in \mathbb{R}^n$represents a vector of n observable values that is related to $\vec{p} \in \mathbb{R}^k$, a set of k unknown parameters is the vector consisting of all undetermined parameters, $p_1, \ldots p_k$ (whose values are to be estimated) by the design matrix, $M(\vec{j}_1, \ldots \vec{j}_n) \in \mathbb{R}^{n \times k}$, whose row vectors are functions of $\vec{j}_1, \ldots \vec{j}_n$, sets of variables that can be independently controlled, $\vec{N} \in \mathbb{R}^n$ represents the observation noise vector with elements being random errors, normally distributed, with a mean of 0 and a variance of σ^2.

In many parameter identification/design of experiments situations, one has latitude in selecting the location of the observation/design points. Thus, the problem of selecting appropriate locations and number of design points in the space of $\vec{j}_i$ to get robust estimates of the parameter vector, $\vec{p}$ is the design-of-experiments problem or the problem of designing an observer for the model given in equation 47.

The least-squares unbiased estimator of $\vec{p}$, $\hat{p}$ minimizes the sum of square errors, $\|\vec{e} - M\vec{p}\|_2$. $\hat{p}$ is also the best linear unbiased estimator (BLUE), which can be obtained by,

$$\hat{p} = (M^T M)^{-1} M^T \vec{e} \qquad (47)$$

4.2.3.2 Experimental Verification of Error Model and Parameter Estimation

Data collection

To identify the kinematic error model parameters, measurements of the machine tool are taken. These measurements are collected over the entire 3D space using a Laser Tracker and Active Target system (Figure 36(a)) to ensure that all axis-dependent machine tool geometric errors are captured. The Laser Tracker used in this test is the API Radian which has a static measurement accuracy of +/- 10 µm or 5 ppm (2σ) according to the specifications provided by API. From this and the tracker's position on the machine, the largest measurement standard deviation (σ) value over the measured range was calculated to be 8.9 µm. In order to ensure that the Laser Tracker was thermally isolated from the machine tool, a plastic Isolation Block was placed between the Laser Tracker base and the machine tool.

Before measurements are taken, a measurement frame is identified. With the Laser Tracker attached to the machine tool bed and the Active Target attached to the machine tool spindle, as shown in Figure 36(a), the B-Axis is rotated with the other axes stationary in order to generate a circle of points. The normal vector of this circle is used as the vertical (Y-Axis) of the measurement frame. Next, the B-Axis is re-oriented to its 0° position and three points are measured as the machine moves along its X-Axis. The best fit line to these points is used as the X-Axis direction of the measurement frame. A right-handed frame is established from these two axes. This frame is then transformed into the negative Y-Axis direction by the Y-Axis encoder value of the machine tool in order to account for the Y position of the machine tool spindle during the measurement frame identification.

The machine tool repeatability, which establishes the maximum possible accuracy for a perfectly compensated machine tool, was calculated next. To determine the machine tool's repeatability, eight quasi-random points from the machine tool's working joint space were measured ten times each. Each cycle of the eight points was measured in a different randomized order to approximate arbitrary approach directions. The error of each measurement is

$$e_{i,j} = \sqrt{\left(x_{i,j} - \bar{x}_i\right)^2 + \left(y_{i,j} - \bar{y}_i\right)^2 + \left(z_{i,j} - \bar{z}_i\right)^2} \qquad (48)$$

where $e_{i,j}$ is the error of the j^{th} measurement of the i^{th} point, $\left[x_{i,j}, y_{i,j}, z_{i,j}\right]$ is the j^{th} measurement of the i^{th} point, and $[\bar{x}_i, \bar{y}_i, \bar{z}_i]$ is the average measurement of the i^{th} point. From the measurements taken of the machine, the largest error was 0.0217 mm, which is used as the machine tool's repeatability. It should be noted that this repeatability value is only 2.4 times the

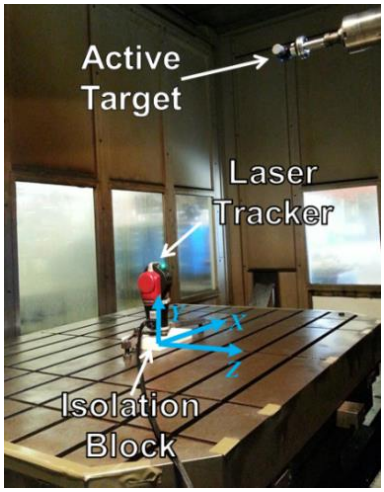
measurement standard deviation meaning that a large portion of this value is may be due to the accuracy level of the laser tracker as opposed to the machine itself. Despite this fact, this repeatability still corresponds to the highest potential measured accuracy of the machine if it was perfectly compensated.

The measurement locations used for model identification and testing were selected next. For the identification set, 290 quasi-random points were selected throughout the machine tool's joint space, and an additional 50 quasi-random points were generated as a testing set. The number of points selected for identification and testing was selected through past experience with similar sized machine tools. This number has the necessary richness to appropriately identify the geometric errors of the machine tool while minimizing the machine tool's down time. The joint ranges used to generate these points are shown below in Table 1, and the distributions of the points are shown in Figure 36(b).

Table 10 **Minimum and maximum commands used for modeling and testing.**

Axis	Minimum Command	Maximum Command
B	0°	360°
X	-1250 mm	1250 mm
Z	900 mm	2200 mm
Y	350 mm	2500 mm
W	-800 mm	-200 mm

(a)



(b)

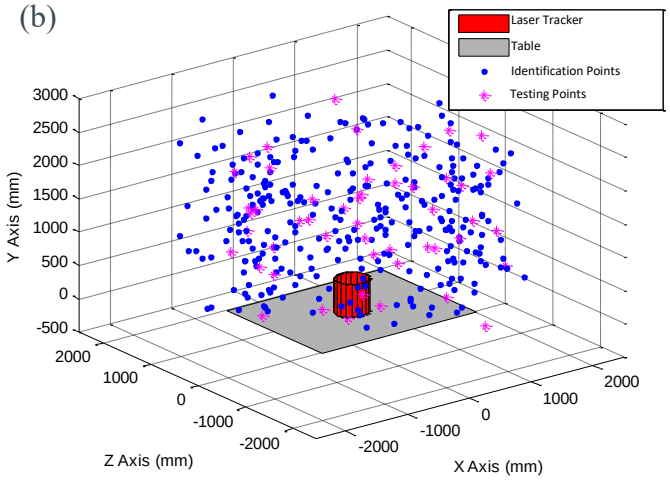


Figure 36 **Machine tool work cell and table base frame; (b) Positions of identification and testing points inside working envelop, given in machine tool base frame.**

Using the Laser Tracker and Active Target system, the 290 point identification set was measured twice. In each measurement set a different length mount was used to attach the active target to the spindle as shown in Figure 36(a). Because the rotation of the spindle does not need to be modeled, these two mounts (Figure 37(a)) allow for the spindle orientation to be determined for each point by finding the vector between the measurement sets.

Because the same axis commands are used when taking both sets of identification points, it is possible to use the two sets of measurements to examine the potential existence of thermal drift in the measurement setup. For each point in the identification set, the distance between the two measurements of that point is ideally equal to the tool length difference of the two Active Target mounts (within machine tool repeatability). Therefore, if the distance between measurements is larger than the repeatability (0.0217 mm), then some shift must have occurred during the time that the system was measured. The distances between the measurements from each set (with the tool length offset removed) are shown in Figure 37(b). The distance between corresponding points ranges from -0.13 to 0.11 mm. Since this value is approximately six times the measured repeatability value, there is evidence that some sort of drift occurred during the measurement process. Furthermore since the air temperature changed by 3.7°C during the measurement process, thermal effects is a likely source of some or all of this drift.

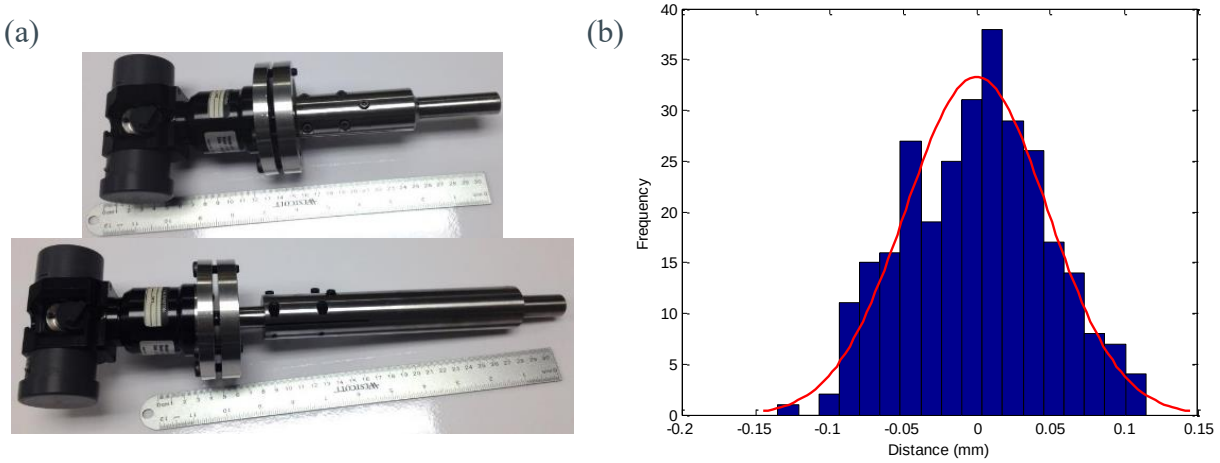


Figure 37**(a). Active Target machine tool spindle mounts; (b) Distance between short tool and long tool measurements.**

Best-fit measurement frame

The procedure described previously was used on the data collected in both the identification and testing data sets. Table 11 shows the estimated errors between the nominal measurement frame and the machine's reference. Also shown in the table is the mean magnitude of the residual error vectors at the measurement points. For identification purpose, two sets of measurement were taken using different lengths of tool. After that, the identified parameters were used for modeling the testing sets.

Table 11 **Best measuring frames of each measuring set.**

Set	Short Tool	Long Tool	Test 1	Test 2
x_0(mm)	0.00845	0.0122	0.00715	0.0111
y_0(mm)	0.351	0.304	0.293	0.298
z_0(mm)	-0.0147	-0.0124	0.0095	0.0116
α_0(rad)	-6.21E-06	3.68E-05	5.78E-05	6.68E-05
β_0(rad)	-2.13E-05	-1.85E-05	-2.43E-05	-2.35E-05
γ_0(rad)	1.05E-05	2.41E-05	3.52E-06	5.05E-06
Res(mm)	0.4214	0.3175	0.2783	0.2745

Parameter identification

The results of parameter identification are shown in Table 12. The data for the two different tools (short, 312.035mm, and long, 435.185mm) were analyzed separately to identify the error parameters of the two identification sets. From Table 12, the high correlation between the parameters identified in the two experiments is apparent. The deviations seen are due to the temperature changes between the two experiments and the uncertainty in the assembly of the target on the tool. Figure 38(a) shows the distributions of the residual errors. The statistical analysis of the results is shown in Table 13. Compared with the residual errors obtained from the frame alignment process, the error model reduces not only the mean but also the maximum (which characterizes the worst-case uncertainty of the machine/model) errors by 90% and 82% respectively.

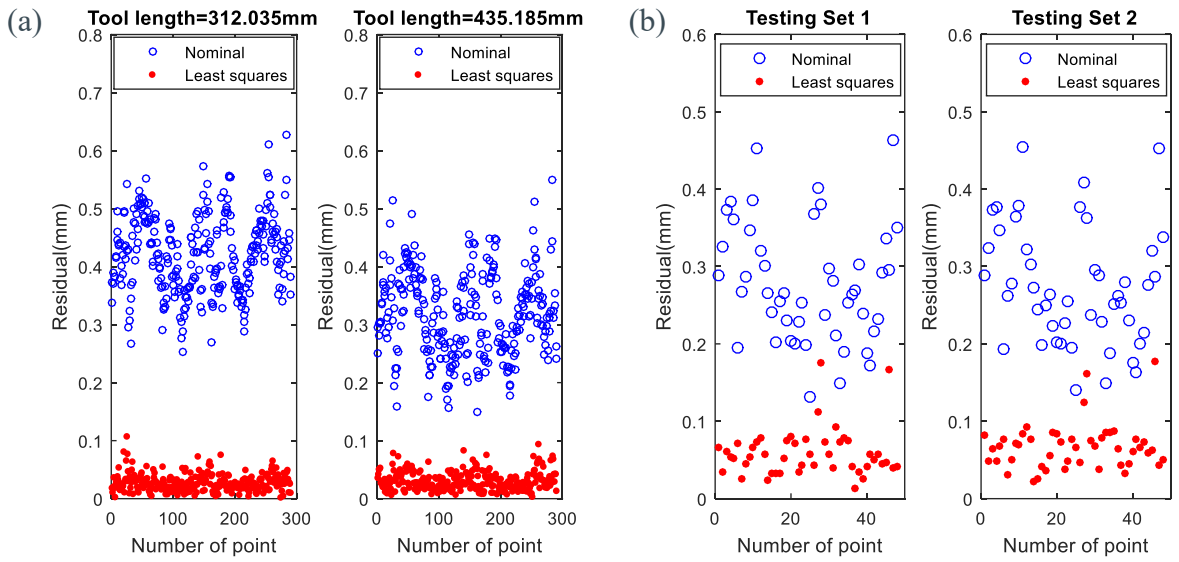


Figure 38 **The magnitudes of error residuals on two identification sets (290 points in each); (b) the magnitudes of error residuals on two testing sets (48 points in each).**

Table 12 **Values identified for the parameters of the error model.**

Unit: mm			Unit: rad			Unit: rad/mm		
Parameter	Short	Long	Parameter	Short	Long	Parameter	Short	Long
$x_1+...x_5-t_l\beta_5$	3.62E-02	5.92E-02	α_1	-2.36E-05	-1.96E-05	$d\alpha/dx$	1.55E-08	2.57E-08
$y_1+...y_5+t_l\gamma_5$	-4.78E-02	-2.35E-02	$\alpha_2+\alpha_3$	-1.01E-05	-1.18E-05	$d\beta/dx$	1.13E-09	7.01E-09
$z_1+...z_5$	-3.12E-01	-1.97E-01	β_1	-6.93E-05	-6.95E-05	$d\gamma/dx$	-1.04E-08	-1.16E-08
dx	1.13E-02	3.75E-03	β_2	4.16E-05	3.51E-05	$d\alpha/dy$	-1.26E-08	-1.80E-08
dy	2.62E-02	2.71E-02	$\beta_3+\beta_4$	5.83E-05	4.82E-05	$d\beta/dy$	2.57E-08	3.03E-08
dz	-1.28E-02	-7.25E-03	$\gamma_1+\gamma_2$	-1.85E-04	-1.77E-04	$d\gamma/dy$	-5.54E-09	-1.54E-09

Unit: dimensionless			γ_3	9.02E-05	8.19E-05	$d\alpha/dz$	-1.68E-08	-1.34E-08
Parameter	Short	Long	γ_4	-1.85E-04	-2.07E-04	$d\beta/dz$	-3.81E-08	-2.73E-08
δx	-1.23E-04	-1.12E-04	β	-9.57E-07	-9.72E-06	$d\gamma/dz$	2.97E-08	2.78E-08
δy	-1.10E-04	-1.09E-04	α	1.15E-06	6.78E-07	$d\beta/dw$	2.93E-08	1.80E-08
δz	-3.75E-05	-3.75E-05	γ	8.03E-06	4.84E-06	$d\gamma/dw$	-4.10E-07	-4.19E-07
δw	-1.06E-04	-1.06E-04						

Table 13 **Model performance for two sets with two different tool lengths.**

Tool length=312.035mm	Mean Residual	% decrease	Max. Residual	% decrease
Nominal	0.4214 mm	N/A	0.6270 mm	N/A
Least squares	0.0277 mm	93.43%	0.1073 mm	82.88%
Tool length=435.185mm	Mean Residual	% decrease	Max. Residual	% decrease
Nominal	0.3175 mm	N/A	0.5492 mm	N/A
Least squares	0.0307 mm	90.34%	0.0941 mm	82.86%

Model testing

With the error model parameters obtained from the identification sets, the model's prediction capability is checked against two testing sets consisting of 48 previously-unseen data points, taken with the long tool. The results of this testing are shown in Table 14 and Figure 38(b). Compared with the nominal machine errors, the model can provide, approximately, a 75% reduction of average magnitude of errors vectors at the points in the data sets.

Table 14 **Model performance for two testing data sets (Tool length=435.185mm).**

Testing set 1	Mean Residual	% decrease	Max. Residual	% decrease
Nominal	0.2783 mm	N/A	0.4624 mm	N/A
Least squares	0.0590 mm	78.80%	0.1760 mm	61.94%
Testing set 2	Mean Residual	% decrease	Max. Residual	% decrease
Nominal	0.2745 mm	N/A	0.4546 mm	N/A
Least squares	0.0670 mm	75.59%	0.1767 mm	61.13%

4.2.3.3 Summary and Conclusions for Error Modeling and Parameter Estimation

A kinematics model for a 5-axis machine tool with a redundant linear axis has been developed. This model introduced 52 parameters, linked to the error kinematics of the machine-tool, which would need to be identified. Analysis of the model shows that only 32 of them have linearly independent effects on the volumetric errors in the workspace. Procedures for least-squares identification of the error model parameters from observations of the volumetric errors at points in the machine's workspace are also developed.

A laser tracker was used to make measurements at 290 randomly generated points in the machine's workspace. These measurements were repeated with tools of two different lengths characterizing the behavior of the machine with long and short tools. The error model parameters were estimated for these two different data sets. In spite of some thermal drift on the machine between the experiments, the error model parameters estimated remained consistent in both magnitude and sign. Further, the model was able to reduce the errors at the observation points to about a third of their original values. The model was tested on two data sets of 48 observation points each. A similar model performance was observed. The proposed model could be used for error prediction on commanded positions.

The average magnitude of residual error vectors in the training sets were about 30 microns. This is consistent with the repeatability of the machine and the fact that the thermal environment changed during the experiments. The modeling approach, along with the convenience of observing errors as a large set of randomly selected points in a machine's workspace with a laser tracker can make for a very effective means of regularly updating compensation tables of machines.

For better model performance, a thermally stable environment would be necessary. Additionally, tracking the thermal drift of the machine with time would yield better model performance. For this, a quicker (consisting of fewer and more strategically-chosen points) and more convenient

data-collection cycle that can be easily embedded into the normal operation of the machine is needed.

4.2.3.4 Design of Optimal Error Observers of Tracking of Thermal Errors

The construction of the volumetric error model is a commonly-used approach of calibration for a machine tool. As we have seen the preceding discussion, depending on the size, design and complexity of the machine, a volumetric error model is built with several unknown error parameters whose values must be estimated before the model is used for compensation. We have also seen that this estimation is accomplished by making observations of volumetric errors in the machine's workspace and using a least-squares fitting procedure. To ensure the observations carry sufficient information to make accurate estimates of the values of the unknown parameters, a large number of observations, (quasi-) randomly distributed across the workspace, are required (in the previous section we used about 300 observations). This enforced redundancy in the observations leads to long measurement times, even with automated devices like laser trackers. As a result, it limits the use of such a volumetric error calibration approaches to static, base-line machine-tool calibrations, not addressing the changes that may occur as a result of thermal variations during the operation of the machine. In this section we develop an approach to reduce the number of observations needed for parameter estimations, while still attempting to get robust parameter estimates.

Observer Design for Linear Systems

Data-driven approaches require large amounts of observations during the operation of the machine. Lengthy measurement processes are not possible because of the transient nature of a machine's thermal state. Therefore, even in the support of using a laser tracker, the concept of optimal design of observations is important to identify most informative observations and shorten the length of the time interval for measurements.

As the Gauss-Markov theorem states, the variance associated with BLUE, given by the variance-covariance matrix is minimized for the design characterized by M,

$$Var[\hat{p}|M] = \sigma^2(M^T M)^{-1} \qquad (49)$$

where $M^T M \in \mathbb{R}^{k\times k}$ is called the information matrix.

The variance-covariance matrix captures the uncertainty in the correlation between the elements of the estimator, $\hat{p}$. It must be noted that σ^2 is the variance of the residual, a property of the random process. So, the uncertainty in the values of the parameter vector and the predictions they make can be seen to be completely dependent on M. As a result of the above considerations, a number of optimality criteria associated with different matrix norms of the design matrix, M have been proposed in both the design on experiments (DOE) and the design of observers (for continuous/on-line estimation and compensation). For a linear regression

design problem with n observations and k unknown parameters to be determined ($n \geq k$), the optimal design represents the selection of n observations that carry the largest amount of information and the correspondingly, the estimator has the lowest variance.

A D-optimal design seeks to maximize information carried by the observations and quantified by the determinant of the information matrix. It does so by minimizing the volume of the confidence volumes or the uncertainty region around the estimator and its predictions. D-optimality is the most commonly used criterion because the target function to be minimized is simpler than the other criteria. Similarly, an A-optimal design seeks to minimize the average variance. The K-optimality criterion which seeks to minimize the sensitivity of estimator to observation/measurement error does so by minimizing the condition number of the design matrix denoted by $\kappa(M)$ which is always greater or equal to 1. It implies that error in observation always corrupt the estimation. The condition number can be infinity if (and only if) M does not have full column rank. $\kappa(M)$ is defined by the worst-case relative errors caused by the error in observations,

$$\frac{\|\Delta\vec{e}\|}{\|\vec{e}\|} \leq \kappa(M)\frac{\|\Delta\vec{p}\|}{\|\vec{p}\|}, where \tag{50}$$

$$\kappa(M) = \frac{\sigma_{max}}{\sigma_{min}}, \tag{51}$$

where $\vec{e}$ is the accurate observation, $\Delta\vec{e}$ is the errors in observation, $\vec{p}$ is the correct estimation, $\Delta\vec{p}$ is the error in estimation due to $\Delta\vec{e}$ and σ_{max} and σ_{min} are the largest and smallest singular values of M respectively.

D, A and K-optimality criteria are all related to the eigenvalues of the information matrix. All three types of design problems deal with the maximization of information, quantified by surrogate functions of these eigenvalues. In this paper, K-optimal design is selected to reject the measurement noise. However, the optimal design theories produce the best locations for observations to identify model parameters under the assumption that the form or degree (if it is a polynomial) of the underlying model of the linear system is known. In many situations, the functions used for machine-tool error models are simplifications (typically with polynomials of axial displacements). Further, to keep the number of parameters manageable, they are assumed to be low-order polynomials. In such cases, there is always a possibility that neglected higher-order terms may be significant. Any observer design process must take steps to alleviate the deleterious effects of model inadequacy. For example, if one tries to fit a straight line model to a parabolic function, $y = x^2$ over the domain [0, 1] with four observations. The best fitting problem is given by,

$$\begin{bmatrix} y_1 \\ \vdots \\ y_4 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_4 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = M\vec{p}, \tag{52}$$

where $x_1, \dots x_4$ are the positions of the observations, $y_1, \dots y_4$ are the corresponding observations and M is the design matrix.

The K-optimal design problem is given by,

$$\min_{\vec{j}_1 \dots \vec{j}_n} \kappa(M) \ni \vec{j}_i \in \Gamma(i = 1, \dots n), \tag{53}$$

Where $\kappa(M)$ is the condition number of M and Γ is the design space.

In this example, $\Gamma = [0,1]$, $\mathrm{n} = 4$ and $\vec{j}_i = x_i$ for i=1,…4. The K-optimal design suggests that the best four observations for equation 53 are x = 0, 0, 1 and 1, so the line fitted by these observations is $y = x$. The corresponding $\kappa(M)$ of these four observations is minimized to be 2.618. It's been observed in Figure 39 that the straight line defined by the end points only has good model performance at two ends. However, the best linear fitting of $y = x^2$ over $x \in [0,1]$ that minimizes the sum of squared error is $y = x - 0.1667$, the green line in Figure 39. The observers produced by the optimal designs localize the observations at the boundaries of the design space, which causes the poor overall fitting performance. To avoid localized observation points, one can introduce constraints to the optimization procedure so as to distribute observations over the domain or design space. For example, the distribution can be one observation between 0 and 0.25, two between 0.25 and 0.75 and the last one between 0.75 and 1. A generalized constrained K-optimal design problem is given by,

$$\min_{\vec{j}_1 \dots \vec{j}_n} \kappa(M) \ni \vec{j}_i \in \Gamma_i(i = 1, \dots n), \tag{54}$$

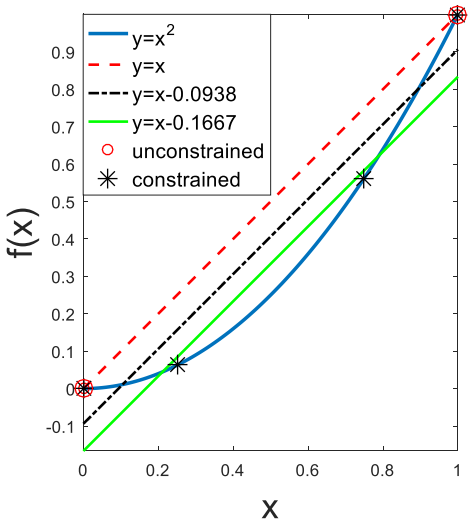


Figure 39 **Quadratic function fitted by linear function**

Where $\kappa(M)$ is the condition number of M and Γ_i are the constrained regions (n=4, $\vec{j}_i = x_i$, $\Gamma_1 = [0,0.25]$, $\Gamma_2 = \Gamma_3 = [0.25,0.75]$ and $\Gamma_4 = [0.75,1]$ in this case).

The solution to equation 54 gives four different observers, x=0, 0.25, 0.75 and 1. The corresponding $\kappa(M)$ of these four observations is 3.25, which is larger than the unconstrained counterpart, 2.618. However, the line fitted by the observers of the constrained optimization is much closer to the best fitting line, $y = x - 0.1667$. Thus the judicious introduction of constraints to obtain distribution of the points balances the need to maximize the amount of information in the observer design with the need to guard against inadequacy of the proposed model. In this example, one might realize that the minimum number of observation points required for estimating the model parameters is two. By introducing redundancy (two additional observations) and constraining the locations of these extra points, one can provide the optimization procedure the flexibility to maximize the information content while, at the same time, ensure that all regions of the domain of the fit are represented. We use this strategy in the design of error observers for machine tools.

4.2.3.5 Observer Design for the Errors of a 5-axis Machine

A schematic of the 5-axis machine used in this study and its kinematic equivalent. The terms in can be rewritten so that we have $\vec{e}(\vec{j}) = M(\vec{j})\vec{p}$, where $\vec{e}(\vec{j})$ is the error vector at a point at some observation point $\vec{j}$. $M(\vec{j})$ are row vectors whose terms are functions of the machine's axial displacements at point $\vec{j}$, and $\vec{p}$ is a set of unknown parameters (the perturbations introduced to the ideal kinematics of the machines, for example yaw of the x axis, rate of change of yaw error with displacement of the y-axis, rate of accumulation of positioning error of the w axis, etc.) It was found out that for the machine described above, the unknown parameter vector contains 32 elements.

The concepts discussed in last section were tested on the machine and error model whose construction was discussed in the previous paragraph. The volumetric errors were observed by a laser tracker. 290 quasi-random observations in the machine's workspace were taken to identify 32 parameters in the linear error model with modeling residual of 27.7 microns. The model was also tested to be valid on a different and smaller data point set. The linear error model in equation 55 can be rewritten as the form of design matrix times the error parameter vector, $\vec{p} \in \mathbb{R}^{32}$.

$$\begin{bmatrix} e_{x,i} \\ e_{y,i} \\ e_{z,i} \end{bmatrix} = \begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} \vec{p} + \vec{N} \cong M_i \vec{p}, \qquad (55)$$

where $M_i \in \mathbb{R}^{3\times 32}$ has elements that are functions of commanded X, Y, Z, W and B axes positions, $\vec{N} \in \mathbb{R}^3$ is the observation noise vector, and the predicted volumetric error components are given by $e_{x,i}$, $e_{y,i}$ and $e_{z,i}$.

To track the evolution of thermal errors, based on consultation by the users, it was decided that we design the observer so that the time for making measurements was limited to 25 minutes. Based on prior experience, it was therefore decided to limit the number of measurement points

for the observer to 80. The linear error model in equation 56, as previously explained can be rewritten in the form of design matrix times the error parameter vector, $\vec{p} \in \mathbb{R}^{32}$. With 80 observations, an identification system with 240 equations can be produced, given by:

$$\vec{e} = M(\vec{j}_1 \dots \vec{j}_{80})\vec{p}, \tag{56}$$

Where $M \in \mathbb{R}^{240\times32}$ is the design matrix controlled by 80 design points $\vec{j}_1 \dots \vec{j}_{80}$ and $\vec{e} \in \mathbb{R}^{240}$ contains the components of the measured error vectors in the observer point-set.

The optimization problem that seeks to maximize the amount of information carried by 80 design points suggests the best set of axes command. Each design point is controlled independently by the commands of X, Y, Z, W and B axis. Therefore, the problem has 80x5 degrees of freedom subjected to the size of command space defined by the limitation on each axis.

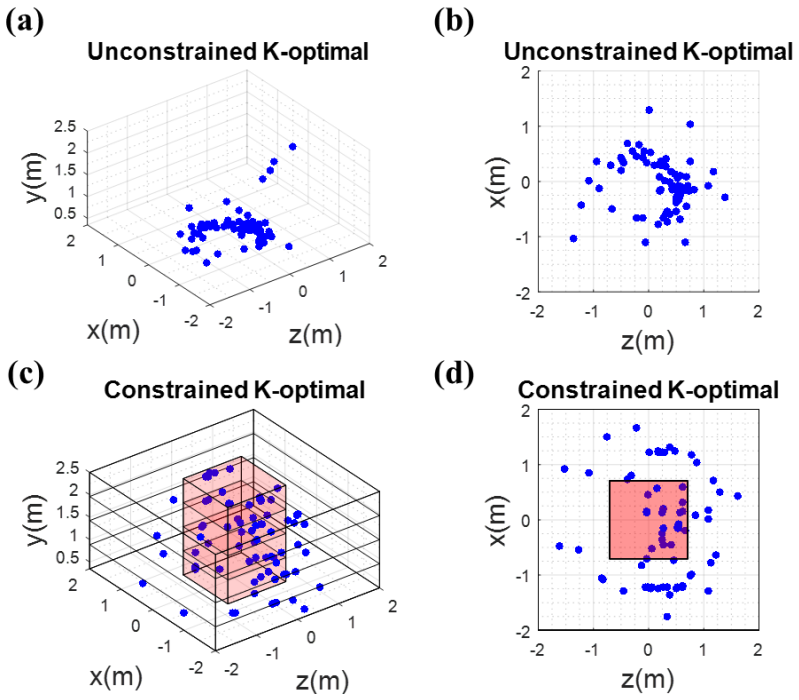


Figure 40 **K-optimal designed observers given in machine tool base frame**

The K-optimal observers can be produced by solving the constrained optimization problem. This was encoded in a MATLAB program, using the generalized constrained optimization function FMINCON takes as input the definition of M in terms of axial positions of the machine and the constraints of the workspace. This function finds a local minimum, hence it was called several time with different randomly generated, feasible initial solutions. In all cases, the target function converged to the close values. The positions of the measurements points in the work envelop of the machine are shown in Figure 40(a). The condition number of the unconstrained K-optimal designed matrix is greatly reduced to 111.3, compared with 437.8, the condition number of the

design matrix given by 290 quasi-random observations. However, the observations for the K-optimal observer are highly localized and located primarily near the bottom of the workspace. In the process of minimizing the influence of errors in observations, it also minimizes the influence of the observations. As aforementioned, the unconstrained location of points for the optimal design of observers is expected to produce such localization. To obtain a more uniform location, the workspace of the machines is sliced into 4 zones along the y axis. Each slice is further decomposed into a central block and an annular space (having the same volume). Thus, the workspace is broken up into 8 equal volumes. Constraints are then introduced into the aforementioned optimization program to ensure that each of these 8 volumes contains 10 measurement points of the new "constrained" K-optimal observers. The result of the introduction of these distribution constraints is given in Figure 40(b) the condition number for the constrained K-optimal design matrix is 207.3. It can be seen that the constraints successfully spread the measurement points over the whole workspace, but the price paid for introducing these constraints is also apparent in the increment of condition number.

4.2.4 Users & Use Cases

4.2.4.1 Experimental study of the behavior of the Constrained K-Optimal Observer

An experiment was designed to test the aforementioned 80-point, constrained K-optimal observer on the machine described in the previous section. We also performed a similar experiment with the unconstrained K-optimal observer. In this experiment, our objectives were to (1) check how error models using the parameter estimates it produces compare with those using parameters estimated from the more traditional, measurement-intensive quasi-random point sets, and (2) determine its ability to track changes in these parameters as the thermal state of the machine changes, and (3) assess improvements, if any, in the observer's performance brought about by the introduction of constraints to distribute the measurements in the workspace.

An API Radian laser tracker with active target system shown in Figure 41 is used to collect the data over the entire 3D space. The 80 points in the K-optimal observer were analyzed for reachability by the tracker. It was found that 4 points were not reachable. The value of objective function for the observer with the remaining 76 points increased from 111.3 to 122.0, which was not changed significantly. The machine was programmed to carry the active target of the tracker and dwell for a few seconds at these 76 measurement points. The tracker and machine were synchronized so that the tracker recorded the position of the target after the machine had settled at a measurement point.

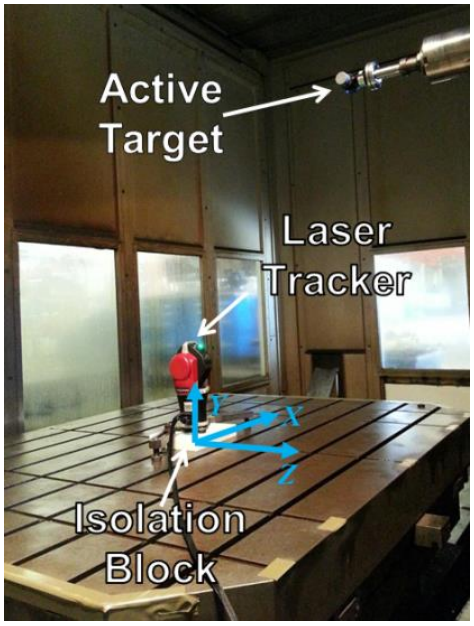


Figure 41 **Machine tool work cell**

This measurement cycle was repeated at intervals of one hour. Six such measurements cycles were performed. The first one at the start of the experiment is able to identify the errors of the machine's initial state. In the four intervals between the first five measurement cycles, the spindle of the machine and the axes of the machine were exercised at roughly half their maximum speeds to heat up the machine. The machine was allowed to cool in the interval between the 5th and 6th measurement cycles. The schedule of the experiment is shown in Figure 42.

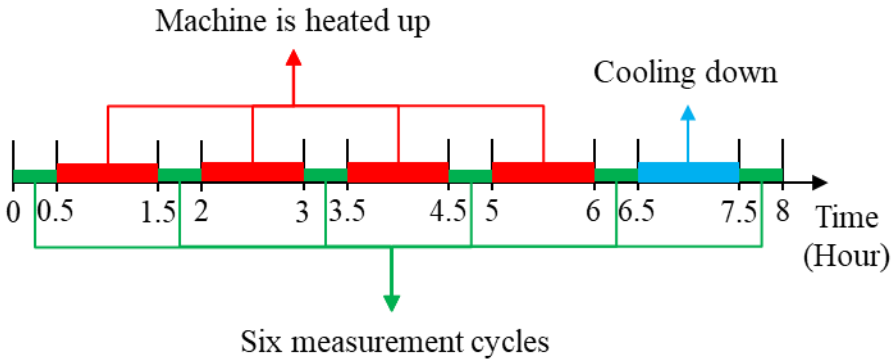


Figure 42 **Measurement, heating and cooling cycles**

Prior to the start of the experiment, the laser tracker is mounted at the center of the table on top of a thermal isolation block. A set of measurements are taken and processed to align the measurement frame to the machine's coordinate system. Further, measurements are made to assess the repeatability of measurements of the laser tracker on the machine. This was found to be around 20 microns. Additionally, the machine was instrumented with 16 wireless thermal sensors to record temperatures at different positions of the machine structure. Four packet radio

transceivers, Adafruit Feather M0 RFM96 LoRa Radio (433 MHz) with 13 temperature sensors (TMP 36) were placed over the course of the experiment, and the data acquisition system transmit temperature data to a computer-based server, which allowed us to monitor and record the temperature in real time. Each of the 4 linear axes was instrumented with 3 temperature sensors (one on the drive and the others distributed around the length of the axes). One of the sensors on the Y-axis was used to monitor the spindle housing temperature. Temperatures were recorded at 1 minute intervals during the experiment.

The experiment was commenced in the morning and concluded late afternoon. The data recorded in each measurement cycle at the measurement points was fed into a MATLAB program and used to identify the parameters of the error model. A similar experiment was conducted with the unconstrained K-optimal observer. In this case, the experiment was conducted without running the spindle between measurement cycles (the reason was to reduce the uncertainty in repeated mounting and dismounting the active target).

The following are some key points in the processing of the data obtained in each measurement cycle. For the initial measurement cycle, misalignment between the measurement and movement frame,$T_{0,1}$ and error parameters, $\vec{p}_1$ are identified separately. In all subsequent cycles, the workspace drift is picked up by the constant terms of the error model. Thus, in the first measurement cycle, we solve two minimization problems:

(1) Identify misalignment between the measurement and movement frame, $T_{0,1}$ in the initial cycle ,

$$\min_{T_{0,1}} \sum_{i=1}^{76} \left\| T_{0,1} H_i \vec{r}_t - \vec{q}_{i,1} \right\|^2, \tag{57}$$

where $T_{0,1} H_i \vec{r}_t$ is the ideal position of the ith measurement point predicted by the ideal forward kinematics and $\vec{q}_{i,1}$ is the actual position measured by the laser tracker at the ith measurement point

(2) Estimate the 32 error parameters, $\vec{p}_1$,

$$\min_{\vec{p}_1} \sum_{i=1}^{76} \left\| M_i \vec{p}_1 - \left(T_{0,1} H_i \vec{r}_t - \vec{q}_{i,1} \right) \right\|^2, \tag{58}$$

where $M_i \vec{p}_1$ is the modelled error, $T_{0,1} H_i \vec{r}_t - \vec{q}_{i,1}$ is the observed error.

For all subsequent (the 2nd to 6th) cycles, the misalignment between the measurement and movement frame $T_{0,j}$ is not updated. $T_{0,1}$ is used as the starting reference for the thermal drift

of the machine and to give the growth in errors due to thermal effects. Therefore, the j[th] error parameter group denoted by $\vec{p}_j$ is identified using a single-step identification (j=2,3…6),

$$\min_{\vec{p}_j} \sum_{i=1}^{76} \left\| M_i \vec{p}_j - \left(T_{0,1} H_i \vec{r}_t - \vec{q}_{i,j} \right) \right\|^2 \quad (59)$$

The modelling residual at the i[th] observation of the j[th] cycle can be obtained by,

$$RES_i = \left\| M_i \vec{p}_j - \left(T_{0,1} H_i \vec{r}_t - \vec{q}_{i,j} \right) \right\|_2 \quad (60)$$

4.2.4.2 Results and Discussion on Constrained K-Optimal Error Observer

The statistics of the behaviour of the models identified for six different thermal states are shown in Table 15. The observed mean and max are the statistics of the errors observed after the tracker placement error are removed from the tracker readings. The residual mean and max are the statistics of the difference between the observed errors and those predicted by the identified models (one would expect these error statistics if the identified model was used for compensation). Figure 43 shows the temperatures recorded by four wireless transmission systems.

Table 15 **Model performances on different states (constrained K-optimal)**

Machine State	Observed (µm)		Residual (µm)		%Decrease	
	Mean	Max	Mean	Max	Mean	Max
Initial	119.1	265.9	26.3	98.2	77.92%	63.07%
Heating	134.4	279.6	31.5	97.2	76.56%	65.24%
Heating	154.0	310.2	28.8	94.5	81.30%	69.54%
Heating	169.6	311.6	33.9	113.4	80.01%	63.61%
Heating	181.9	331.6	30.9	111.6	83.01%	66.34%
Cooling	157.3	309.4	29.8	101.7	81.06%	67.13%

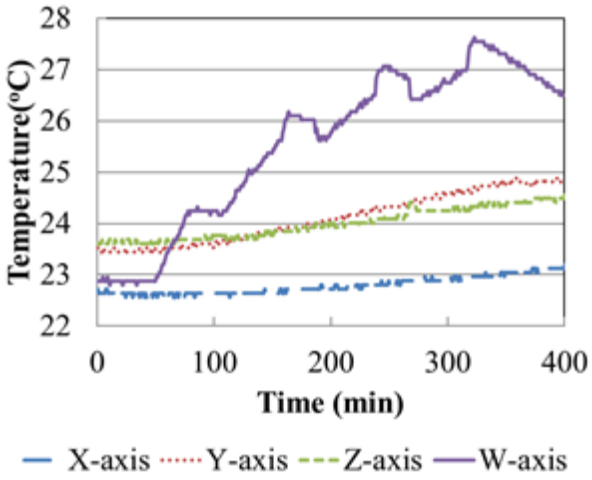


Figure 43 **Temperature variation over the course of the experiment**

The models identified by making measurements at points prescribed by the constrained K-optimal observer over 6 measurement cycles suggest highly repeatable performance for each thermal state. During the first data collecting cycle (initial state), the model provides 77% and 63% reductions in the mean and maximum magnitude of error, respectively. These percentages increased as the average magnitude of the machines errors increased because the average magnitude of the residuals remained a relatively narrow (8 micron) band.

The observer performed well in terms of explaining the error. The repeatability of the positioning was around 20 microns (machine and laser tracker combined). Further, an additional uncertainty of about 5 microns was introduced because the active target had to be removed and remounted between measurement cycles (because the heating cycles required the spindle to be run). The average magnitude of the residuals at the observer measurement points over six experiments was close to 30 microns, suggests that the identified models were capturing most of the systematic errors of the machine and adjusting the parameters appropriately to adjust to thermal changes of the machine.

These results are comparable to those reported in previous work on the same machine, using the same kinematic model but, instead using a quasi-randomly generated set of 290 observation points. In the aforementioned work, the average magnitude of the residuals was 27.7 microns. Thus, with the machine's thermal condition varying, additional uncertainty of removing and replacing the laser target in the spindle and a measurement set reduced by more than a third, the constrained K-optimal observer produced comparable performance. The feasibility of using a smaller and more strategically-chosen point set to perform on-line thermal error tracking is thus demonstrated. The measurement cycle time for measurements for this reduced set of points is only 24 minutes. This suggests that, with a quick data collection strategy and a robust error

model parameter estimation procedure, one might be able to track and compensate the thermal errors as they evolve by executing a process intermittent gaging and error updating strategy.

One can compare the performance of tracking approach to that of a static calibration approach that does not attempt to track and compensate thermal errors. In such a situation the machine is calibrated once (typically, a quarter or a month or, optimistically, at the beginning of a shift) and the results are used, without regards to the thermal state of the machine, for compensation of its errors during operation. Simulating an optimistic situation, where the machine is calibrated at the beginning of the shift and the results of the calibration are used through the entire shift, we would obtain $T_{0,1}$ and $\vec{p}_1$ in the first cycle (initial state) and use these to calibrate the rest of five data sets. Thus, the residual at the ith observation of the jth cycle is given by:

$$RES_i = \left\| M_i \vec{p}_1 - \left(T_{0,1} H_i \vec{r}_t - \vec{q}_{i,j} \right) \right\|_2, (j = 1, \dots 6) \qquad (61)$$

The statistics of the residuals produced by this approach (or the performance of a static compensation approach) are given in Table 16. The average model residual grew from 26.3 to 155.1 microns in the five-hour heating up process and reduced to 120.7 after a one-hour cooling period. Over a 400-minute period of operation, the compensations estimated in the cold state of the machine, though producing some improvements in the error, are seen to become increasing ineffective. After 5 hours of heating, the compensations only produce a 15% reduction in error.

Table 16 **Thermal drifts without updating the error parameters**

Machine State	Observed (μm)		Residual (μm)		% Decrease	
	Mean	Max	Mean	Max	Mean	Max
Initial	119.1	265.9	26.3	98.2	77.92%	63.07%
Heating	134.4	279.6	77.1	122.6	42.63%	56.15%
Heating	154.0	310.2	115.4	180.0	25.06%	41.97%
Heating	169.6	311.6	138.6	208.8	18.28%	32.99%
Heating	181.9	331.6	155.1	235.0	14.73%	29.13%
Cooling	157.3	309.4	120.7	208.6	23.27%	32.58%

In many situations, instead of opting for a static calibration or attempting to update the entire parameter vector (to compensate for workspace drift and distortion), one may opt to probe a few points, estimate the drift and program in a shift of the programming origin based on these measurements. We simulate this situation by doing a full calibration in the first cycle, then we use fixed error parameters, $\vec{p}_1$ but update workspace drift specified by $T_{0,j}$, which now has only three translational degrees-of-freedom. For the remaining five measurement cycles, the workspace drift, $T_{0,j}$ $\mathrm{T_{0,j}}$is identified by:

$$\min_{T_{0,j}} \sum_{i=1}^{76} \left\| M_i \vec{p}_1 - \left(T_{0,j} H_i \vec{r}_t - \vec{q}_{i,j} \right) \right\|^2, (j = 2, \dots 6) \quad (62)$$

The same error parameters, $\vec{p}_1$ is used to predict the errors at the same observer locations. The prediction errors of the ith observation of the jth cycle is given by,

$$RES_i = \left\| M_i \vec{p}_1 - \left(T_{0,j} H_i \vec{r}_t - \vec{q}_{i,j} \right) \right\|_2, (j = 2, \dots 6) \quad (63)$$

As shown in Table 17, updating $T_{0,j}$ is effective in controlling the inaccuracies caused by the thermal effect. The worst mean model residual occurs at the end of the heating period and is measured to be 98.1 microns, which is lower than that produced by using static calibrations. However, the model performance is seen to degrade severely when compared to the full identification of the model parameters. One can see that that using a static compensation and tracking only the drift of the workspace explain only 46.07% of the observed thermal error. These comparisons between full periodic parameter identification, partial (drift only) identification, and no identification not only illustrate the scale of the relative influence of thermal errors (workspace drift and distortion), but also demonstrates the need and importance of periodic updates to calibrations.

Table 17 **Thermal drifts (only compensate the shift of the measurement frame)**

Machine State	**Observed (μm)**		**Residual (μm)**		**% Decrease**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
Initial	119.1	265.9	26.3	98.2	77.92%	63.07%
Heating	134.4	279.6	68.7	184.4	48.88%	34.05%
Heating	154.0	310.2	73.3	203.0	52.40%	34.56%
Heating	169.6	311.6	88.6	243.1	47.76%	21.98%
Heating	181.9	331.6	98.1	255.5	46.07%	22.95%
Cooling	157.3	309.4	84.9	195.6	46.03%	36.78%

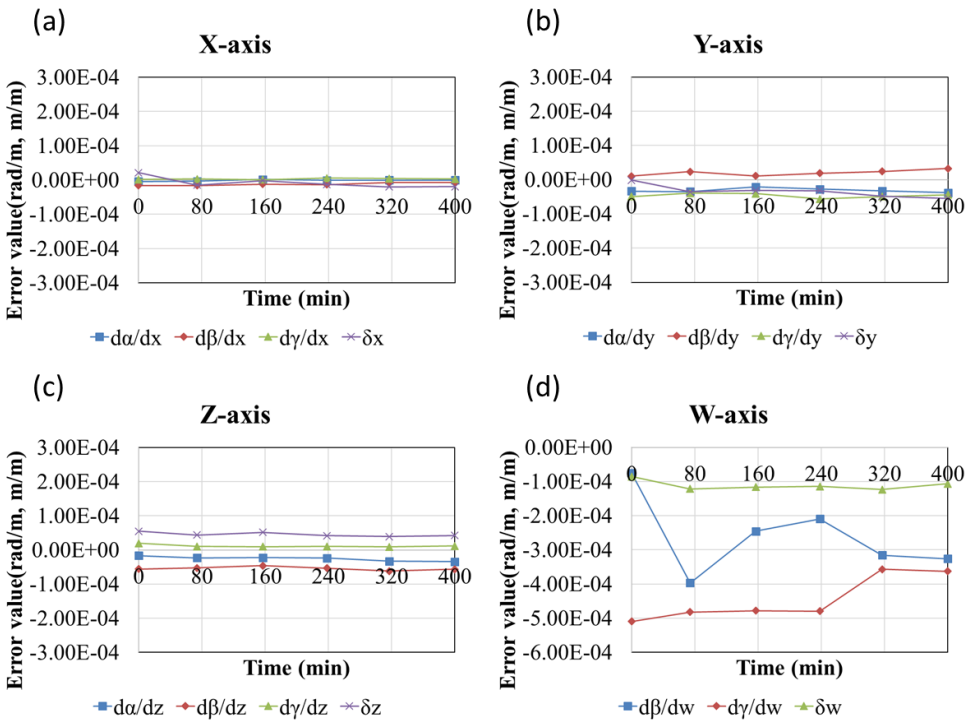


Figure 44 **Variation of error parameters**

The thermal effects are observed to cause the average error of the machine to grow from 119.1 in the cold state to around 181.9 microns after 4 heating cycles. With compensation, the error model parameters (see appendix for descriptions) estimated by the single step identification process error, described in the previous section, would hold the average error of the machine to around 30 microns. The error parameters were identified by the least-squares fitting. The parameters related to X, Y, Z, and W axis at different machine's thermal states are shown in Figure 44. It can be seen that only parameters associated with the Y and W axes show significant changes during four heating cycles. The spindle was turned on during the heating process and the readings of the wireless temperature sensors show that the temperature readings of W and Y-axis increased by 4.5 and 3°C, respectively. The other parameters related to the X and Z-axis, on the other hand, had less than a 2°C rise in temperature during the 400-minute experiment. By studying the thermal behaviour of each axis, one can understand the characteristics of the machine, which might be used in error avoidance. For example, it is observed that the error parameters associated with W-axis are varying significantly during an operation. It shows that the machine's positioning error caused by distortion of W-axis could be more significant. Therefore, the positioning error could be avoided by replacing a W-axis movement with a Z-axis movement. Besides, it is observed that some parameters do not vary significantly over time and could be considered constants (e.g. parameters associated with X and Z axes). By making such assumptions, the number of undetermined error parameters can be reduced and thus reduce the needed number of observations.

As mentioned in the previous section, we also conducted an experiment on the unconstrained K-optimal observer design (i.e., observer obtained without measurement point distribution constraints). The statistics of the results are shown in shown in Table 18. First, because the machine spindle was not run, between measurement cycles, the observed errors stayed constant. The temperatures during the experiment were observed to remain constant to within 2^{o}C. While the parameters identified based on the measurements prescribed by the observer explain about 70% of the observed error, it can be seen that in terms of magnitude, the residuals are higher than those seen in the constrained observer as well as in the quasi-random measurements.

Table 18 **Model performances on different states (unconstrained K-optimal)**

Machine State	Observed (μm)		Residual (μm)		%Decrease	
	Mean	Max	Mean	Max	Mean	Max
Initial	176.0	460.9	47.9	114.5	72.78%	75.16%
Heating	180.4	449.4	49.2	114.8	72.73%	74.45%
Heating	177.2	444.1	48.9	110.8	72.40%	75.05%
Heating	176.7	453.7	48.4	119.5	72.61%	73.66%
Heating	172.9	443.5	47.6	110.8	72.47%	75.02%
Cooling	169.8	427.4	49.8	130.4	70.67%	69.49%
Cooling	163.8	417.6	46.5	105.8	71.61%	74.66%

4.2.4.3 Concluding Remarks on Error Observers

The idea of using error observers for machine tool errors was introduced in this technology. An optimal observer design identifies a set of locations in the machine's workspace at which to make error measurements, so that the information contained in the set to estimate the parameters of a given error model is maximized. The approach can be used for any of the many proposed volumetric/quasi-static machine tool error models. The concept of applying the K-optimal design that minimizes the sensitivity of measurement errors on the parameter estimates has been proposed. The use of a single optimality criterion in the observer design leads to localization of the measurement points either near the center of the workspace or at its boundaries. To overcome the tendency, we introduce constraints, to uniformly distribute the observer points over the workspace. We also introduce redundancy (more measurement points than the minimum needed) to guard against the effects of model inadequacy. Constrained and unconstrained observer designs based on the K-optimal design criterion have been generated for a 5-axis machine.

Experiments have been conducted to assess the behavior of K-optimal (minimizing the condition number of the design matrix) observers. Compared with the condition number of 437.8 for 290 randomly-generated commands, the 80-point K-optimal observers have a better conditioned design matrix with condition number of 111.3. The constrained 80-point K-optimal observer with a condition number of 207.3 is also an improvement. Over six data collection cycles, the

constrained K-optimal observer produced mean and maximum residuals of 30 and 100 microns, respectively, which are comparable to those (27.7 and 107.3 microns) produced by the 290 quasi-randomly generated point set. This clearly demonstrates that a smaller strategically chosen set of measurements can produce estimates comparable to those produced by larger point sets.

In order to test the possibility of using the observer sets to track the thermal drift of a five-axis machine with 32 error parameters, a data collecting cycle consisting of 76 constrained K-optimal observers was used for each of the six thermal states including initial set and four heating and one cooling cycles. This produced six sets of error model parameter estimates. It is observed that the error parameters correspond to W-axis vary the most because it held the spindle. The Y-axis, as the closet axis to W-axis and the second most heated axis, the variation of the error parameters associated with the Y-axis vary is also considerable. The mean and maximum modeling residuals for six thermal states are found to be 26.3 and 98.2 microns, respectively, which are close to the mean and maximum modeling residuals (27.7 and 107.3 microns, respectively) modeled by 290 quasi-random generated points. This also shows that using a smaller observer set does not corrupt the accuracy of the error model. More importantly, the thermal error of the machine was observed to be significant (around 60 microns, from about 119.1 microns to about 181.9 microns over the course of 320 minutes) during the operation of the machine. The largest mean residual error for the six measurement cycles conducted was observed to be 33.9 microns. During this period, if a static compensation model whose parameters were estimated with the machine in a cold state was used, the mean residual error (the average error one would expect after compensation) would have risen from 26.3 to 155.1 microns over the course of 320 minutes. If rudimentary workspace drift was compensated, the residual error would have grown from 26.3 to 98.1 microns. This not only demonstrates that the observer is able to consistently track thermal errors of the machine as its thermal state was continuously varying, but also serves as a reminder of the importance and magnitude the thermal component of quasi-static errors.

This research has demonstrated the feasibility of tracking thermal errors with constrained K-optimal observers with periodic measurements taking only around 24 minutes to perform. Future work will address the evaluation of D- and A-optimal observers. Additionally, faster and less intrusive (than laser trackers) methods for implementing the observers need to be explored. The error model can be simplified by replacing the error parameters with stable thermal behavior with constants. The fewer parameters, the fewer observations would be required to get robust estimation. Finally, this work opens up possibility using temperature readings for tracking thermal errors. By correlating the estimated values of the error model parameter to temperatures in different parts of the machine, it should be possible to compute volumetric errors using only temperature readings, thus reducing the time required by, and invasiveness of, the thermal error tracking system.

4.2.5 Software and Systems Requirements

The construction of error model for a machine, especially machines with more than 3 axis can be complex and might require both algebraic (symbolic) manipulation and computation. MATLAB, which can handle symbolic variables and numerical calculations, is chosen as the environment in which to build software for the techniques developed in this project. After the volumetric error model is derived, it is used for observer design, which is a nonlinear optimization problem with linear constraints. Therefore, several functions are developed and called in the program to set it up and solve it to obtain the optimal observer. These functions are listed below:

1. *x = fmincon(TarFun,x0,A,b)*

Thus function is a solver of the minimum of constrained nonlinear function, which is specified by,

$$\min_x f(x) \ \ such\ that\ Ax \leq b,$$

where $f(x)$ is a nonlinear target function of x given by *TarFun*(explained in the next function), $Ax \leq b$ describes all linear constraints of the problem and x_0 is an initial guess to start the solver.

More details can be found at https://www.mathworks.com/help/optim/ug/fmincon.html.

2. *fx=TarFun(x)*

The function specifies the target to be optimized. In this case, it describes the condition number of the design matrix as function of all controllable variables, *x*.

3. *[A,b, Lower_bound, Upper_ bound]=observer_distributor(min_travel, max_travel,sf)*

This function generates upper and lower bounds for the controllable variables to be optimized in *fmincon.* A and b can be directly fed into the optimization solver.

4. *M=Volumetric_error_model(a,b,c,Joint_command,Tool_length)*

The machine's volumetric error model, $\vec{e}(\vec{j}) = M(\vec{j})\vec{p}$ is the linear relation between a set of joint command, $\vec{j}$ and its corresponding errors, $\vec{e}$. This function depicts the matrix, $M(\vec{j}) \in \mathbb{R}^{3\times 32}$ as a function of all joint commands. *a,b,c* are the angular displacement errors of the laser tracker, use zeros for convenience.

4.2.6 Features & Attributes

The quasi-static error modeling approach we've developed has the following features:

1. The modeling approach requires a versatile metrology instrument such as a laser tracker, allowing for a model with a large number of parameters to be identified, thus proving the effectiveness of the approach.
2. Standard tests and calibration procedures like the ASME B5.54 and the ball-bar only calibrates specific trajectories and expose a sub-set of the underlying error sources. These measurements cannot be composed into complete models for volumetric error

compensation. The proposed approach using a laser tracker, on the other hand, is capable of building the volumetric error model of the machine.

3. The modeling approach is general and can be applied to a variety of CNC machines.
4. In the current application, only first order terms are considered for simplicity. A higher order model that better describes machine's error characteristic is another approach to reduce the modeling residual.

The machine-tool observer design has the following features:

1. The general approach of designing the optimal error observer is not limited to the error model used in this report. The same methodology can be applied to design an observer for any error models (e.g., the error models constructed by different perturbed kinematics or the error models for machines with different structures and designs).
2. The approach is agnostic to the technology used to measure/observe errors. It identifies the most informative positions to be observed in the machine's workspace. The robustness in the estimation is maximized no matter what metrology instrument is used for data collection.
3. In its current implementation, 80 points of measurement are used for designing the observer. This introduces redundancy and but provides robustness. The size of the observer could be reduced to 50~60 measurements points to shorten measurement cycle and embed it into the production cycle.
4. In its current implementation, this theory has been used to verify the feasibility of using K-optimal observers. However, D and A-optimal design observers could be also obtained using the programs develop. They might be applied to different machining scenarios.

4.2.7 Modes of Operation

The program is designed to update the machine's quasi-static error model for compensation purpose over time. Our work has not addressed the introduction of compensatory actions during the use of the machine. However, commercial systems from machine vendors such as VCS from Siemens can be used for this purpose.

1. To start a new calibration cycle, a metrology engineer sets up the laser tracker, which takes the error measurements during the measurement cycle.
2. The machine operator commands the machine to move to the positions of observers, while the laser track collects the data. Once finished, the engineer feed the data into program 1, and the volumetric error model can be obtained.

The above two steps can be repeated periodically to keep the track on machine's thermal behavior.

4.3 Adaptive Machining

4.3.1 System overview

In order to identify casting defects such as insufficient stock or poorly mounted parts, a 3D scan of key surfaces of a given part is desired. In-machine laser scanning provides a process which lends itself to automation of the scanning process since the machine itself can provide the motion necessary for the scanning device to view all surfaces of interest on the part. In this project a calibration process was developed suitable for automatic identification of the location and orientation of the scanner head with respect to the machine end effector. A method using VERICUT to lay out and verify desired scan paths was developed along with MATLAB software to generate the g-code and reference files needed by the scanning software. Several methods of linking the scan data with the associated position of the machine corresponding to any given scan line were examined and the limitations of each identified. A hybrid method combining the discrete and constant velocity methods was also developed in order to allow high density data on machines like the PAMA whose limitations prevent the use of simpler methods. Software was also developed to automatically collect scan data from each individual scan pass and transform the point coordinates from each data set into the same machine base frame.

For in-machine scanning of a part, the architecture used is displayed in Figure 45. A trigger signal from the machine tool controller is needed to coordinate the motion of the machine and the collection of scans. If access to the encoder signal for the axes used for scanning this would be ideal, but several methods were developed which are able to make use of any single digital signal that can be triggered with an M code within the NC program.

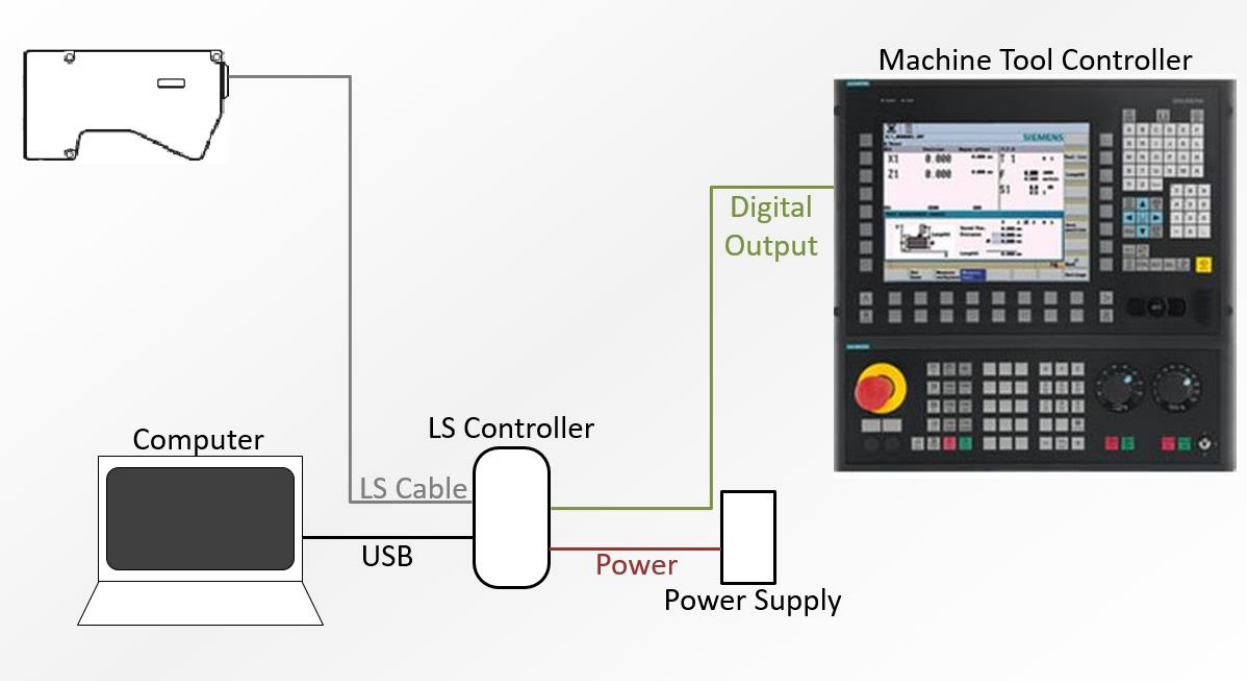


Figure 45: In-machine scanning system architecture (

A virtual gage is a digital simulation that combines real data (measured from an artefact) with a computer representation of a condition or test that the data should satisfy. For example, the

former might be a point-set extracted from the scanned data of a part while the latter might be the equation of a plane extracted for part model that represents an ideal material condition that must be satisfied by all points in the aforementioned point-set. The virtual gage assembles these entities into a common reference frame, creates the appropriate set of inequality conditions to be satisfied, and then checks them on each point in the point-set. Figure 46(a) shows a typical situation encountered in the test case of deciding the acceptability of a casting for final machining. To produce an acceptable finished part, a machining allowance of d_o is desired on each machined surface. The virtual gage software computes the rigid body transform (translation and rotation) of the point-set to produce each point in the set representing the casting surface satisfies the gage equation (given in Figure 46(a)). Obviously, a part will have several such specifications and the virtual gage software is expected to find a single rigid body transformation with which to simultaneously satisfy all of them. One may introduce conditional steps into the procedure, for example, if all the gages can be satisfied, then the software should 'equalize' the allowances on all machined surfaces. As is apparent, each virtual gage introduces one or more sets of inequality constraints as does the conjoining of the reference frames of point-set and gage planes. Thus, the problem becomes a constrained optimization problem, where an optimal solution satisfies the virtual gage constraints and minimizes or maximizers some objective (such as difference in machining allowance on all machined faces).

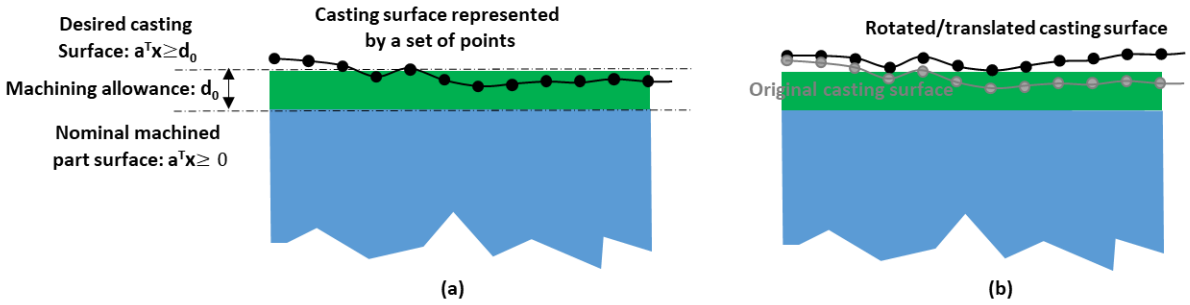


Figure 46 Virtual gage theory

Thus, in summary, to setup a virtual gaging problem, a point cloud representing the physical part and virtual gages idealized geometrical surfaces and representing the dimensional, tolerance and allowance specifications are required. In the GD&T standard, dimension and tolerance are defined based on a reference or datum coordinate frame. This may be different from the coordinate system in which machining is programmed. For simplicity, a homogeneous transformation is applied so that the datum coordinate system is made coincident with the machining programming coordinate system.

The Adaptive Machining system also consists of a software that can be run independently to modify the nominal NC program of the part to be machined by introducing minimal amount of additional lines in the nominal NC program. The software could be run in different configurations to provide the user with different levels of control in modifying the final NC program. As part of this project, we also identified a commercially available simulation software program (Vericut)

that can be used to simulate and verify the modified NC program. In particularly, we updated the CNC machine model and tool definitions for machining simulation. The software also has capability to perform machining simulation using a partial scan data and identifies tool and fixture collision errors, if any, in the process. The system was demonstrated for both nominal NC program modification and simulation.

4.3.2 System Architecture

Overall Adaptive Machining system architecture is show in Figure 47 and the procedure list as below.

1. Incoming part variation is too large to be manufactured without adaptive manufacturing technology, and it needs to be reworked or scraped
2. Scan part to get point cloud of the part's geometric variation and misalignment information
3. Calculate feasible solutions & optimize compensation
4. Generate adaptive manufacturing process with compensation information
5. High-quality manufactured part with all the features that meets the requirements

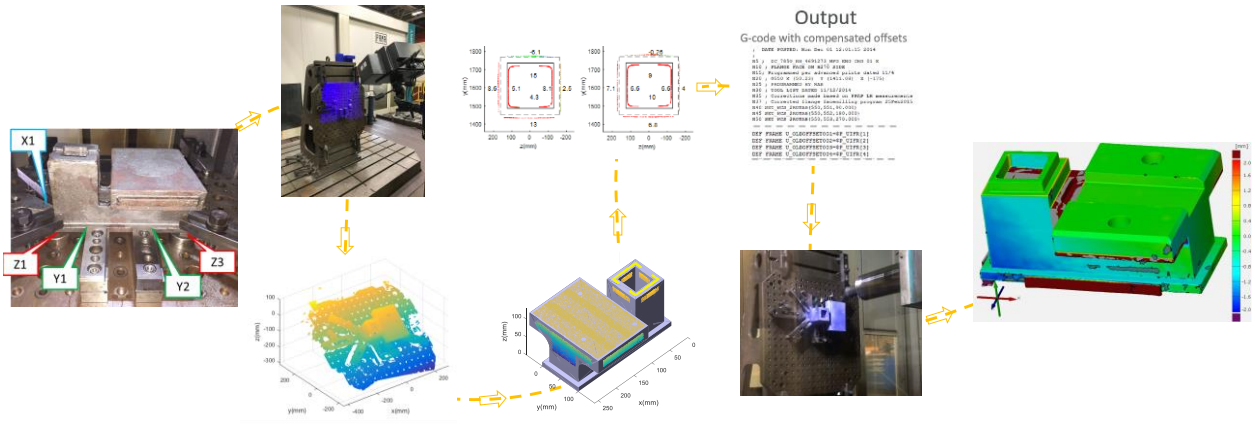


Figure 47 Adaptive Machining system architecture

The structure of virtual gage system operation is shown as a flowchart given in Figure 48.

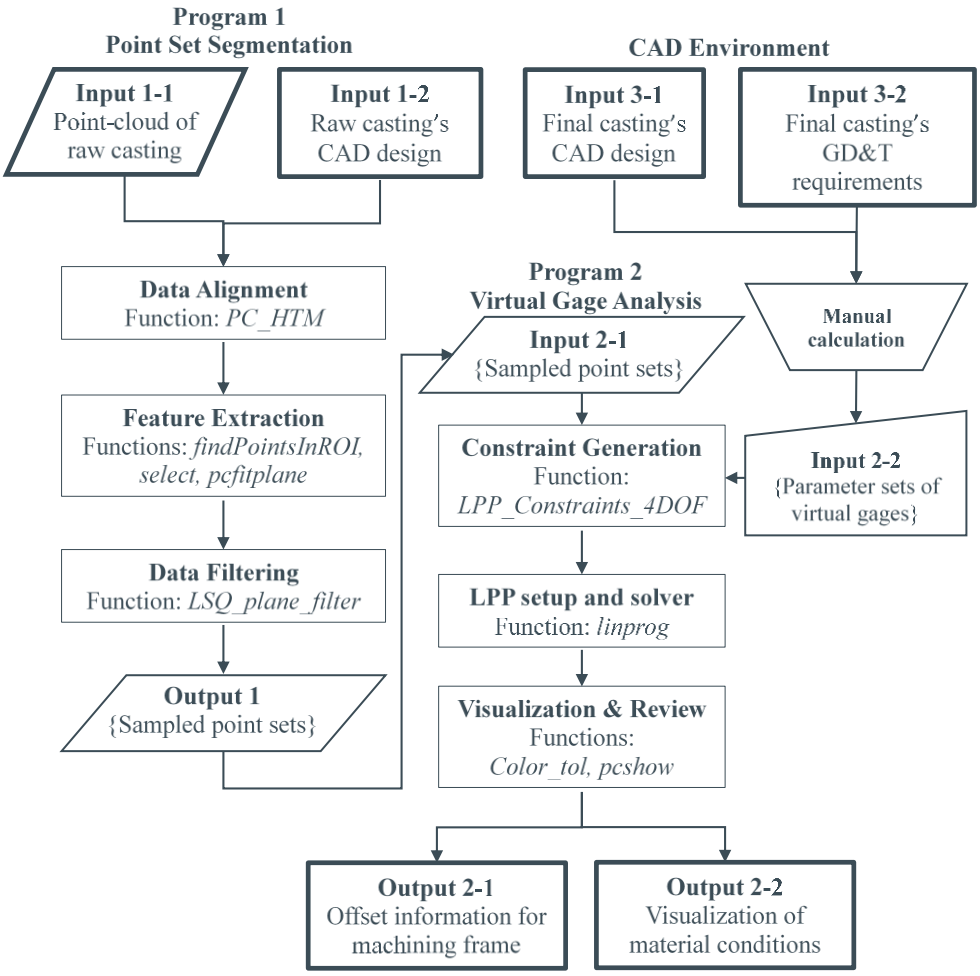


Figure 48 Structure of virtual gage system operation

Figure 49 represents the workflow for the NC program modification and its implementation during the adaptive machining process.

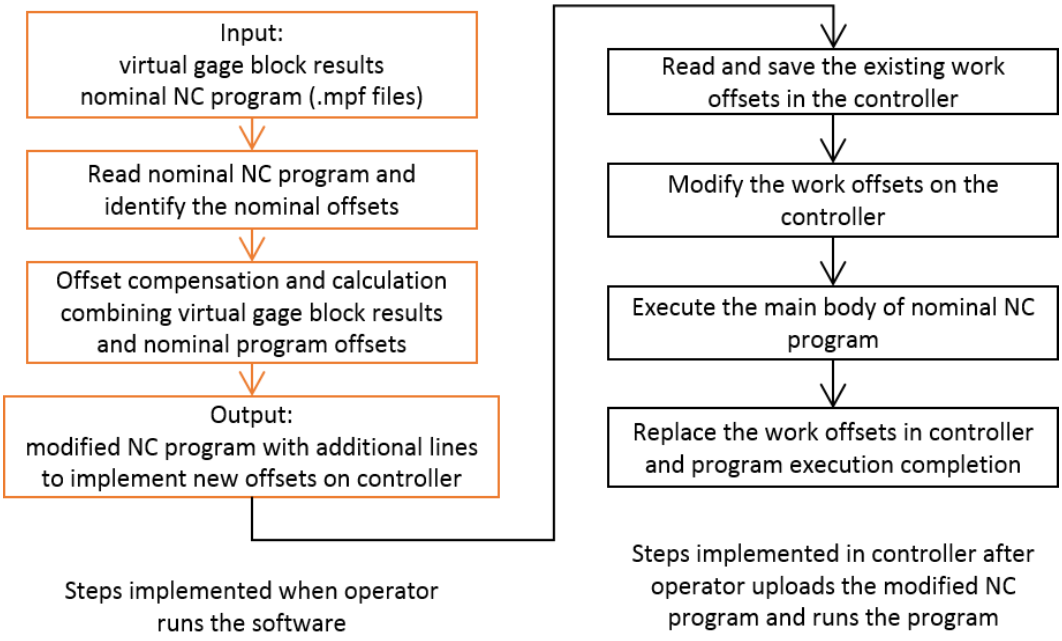


Figure 49 Flow chart showing the implementation of the adaptive machining process

An alternative approach has also been explored in the current project to modify the tool path of the NC program operations in the CAD/CAM module before generating the NC program. This approach is different from minimally modifying the NC program with re-defining offsets as the tool path for different operations is modified in the CAD level. This approach will be applicable for adaptive machining in all cases where the machining operations are created using Siemens NX software. The software for this approach is created in visual basic programing which could be run as a journal program in the CAM module of the Siemens NX software. The workflow is shown in Figure 50.

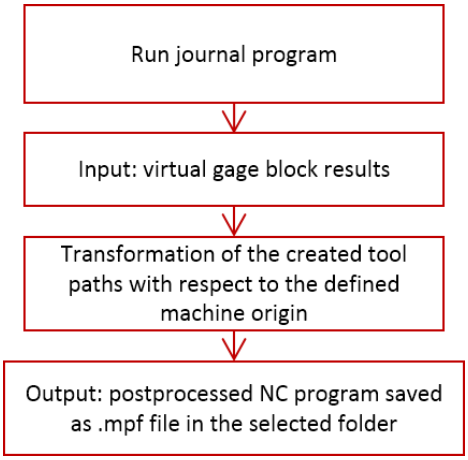


Figure 50 Flow chart showing the workflow for the tool path transformation approach during the adaptive machining process

4.3.3 System Development

In order to accurately transform the raw scan data into 3-dimensional coordinates, it is necessary to know the machine's position when each scan line was captured. Several modes of operation are possible:

Discrete method

This method consists of moving the machine to a desired location and stopping, capturing a single scan line, then moving to the next desired position. The discrete method therefore has exact information about the position of the machine for each scan line. The discrete method is also simple to implement and requires only a digital output from the controller to trigger each scan. The discrete method is however very inefficient with respect to time if a relatively dense coverage is desired, as the machine must stop at each desired scan line location. Thus, the discrete method is only ideal for sparse data collection.

1. *Encoder method*

   The encoder method is ideal if access to the encoder signals is available. In addition to the digital output signal from the controller used to signal the start of each scan pass, the encoder signal lines from the axis moved during the scanning can also be wired to the Laser Scanner Controller. The scans can then be triggered based on encoder pulses which are spaced in known increments from the start position of each scan pass. This method produces exact information regarding the location of each scan line, and is capable of collecting dense data efficiently. The required access to an encoder signal may limit its use. In the case of the PAMA machine, access to the encoder signals without violating the manufacturer support policies would have required a custom service from the manufacturer with cost that was not feasible for the scope of this project. This method was validated on a small 3 axis machine at Missouri S&T and produces results equivalent to the discrete method but only requires the machine to stop between scan passes and can collect up to the full scan batch size limit of 15,000 scan lines per pass with density up to the resolution of the encoder signal.

2. *(Constant) Velocity method*

   The alternative to using an encoder signal to trigger scan line collection is to capture at a set time spacing after the scan batch is triggered to start. Assuming the machine is moving with constant velocity, this will result in a regular capture of scan lines with spacing based on the machine travel speed divided by the scan capture frequency. Figure 51 illustrates the fact that due to differences in communication timing, the machine and scanner will likely not start simultaneously. The machine will also not instantaneously reach the set constant velocity. Therefore there is some unknown offset between the actual machine position and that which would be initially calculated based on the scan line number under the assumption that the scan data starts at the same instant as the machine and are evenly spaced positions. There may also be portions of the scan data at the beginning and end which must be discarded due to not being constant velocity during acceleration and deceleration of the machine. If the timing offset between the machine and scanner start is repeatable (as well as the machine's acceleration profile being repeatable) then

the offset can be identified during a calibration process and applied to any scans thereafter. This method is comparable to the encoder method in data collection efficiency (though the acceleration and deceleration periods at the ends of each scan pass are not valid in this method). The accuracy of the information regarding position of the machine for each scan line depends on the consistency of the start timing of the machine and digital output used to trigger scan batches. For the PAMA machine however it was found that the timing of the machine motion start was not consistent, and therefore the resulting offset was different for every run making this method infeasible.

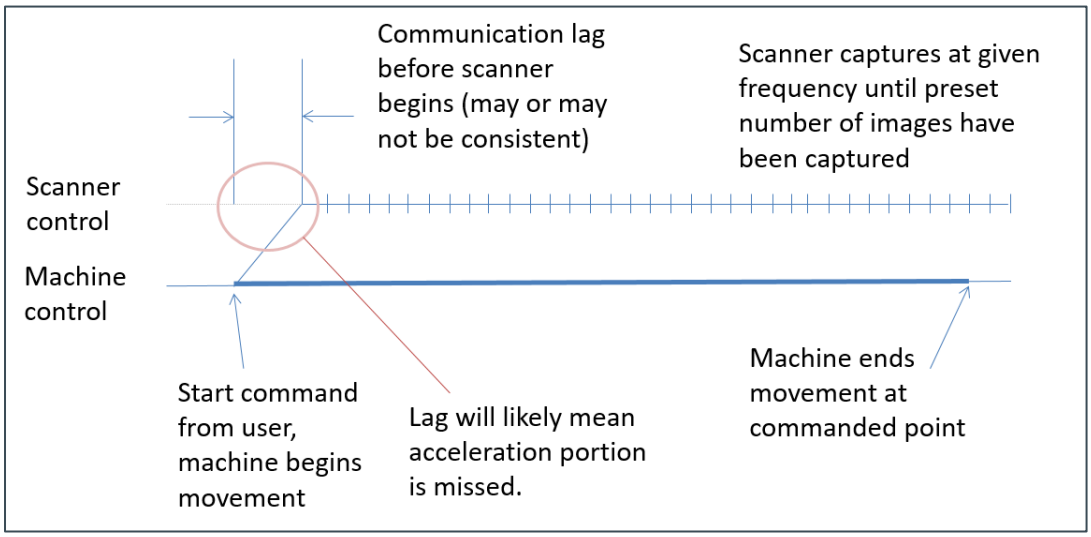


Figure 51: Velocity method

3. *Hybrid method*

This method was developed in an attempt to overcome the limitations of the Velocity method and allow for dense data collection on the PAMA. Scans are collected with the desired data density using the Velocity method and the Discrete method is also used to collect scan lines sparsely distributed through the scan passes used for the velocity method. The offset for each scan pass is then calculated by finding the offset value that best aligns the Velocity data with the Discrete measurements (which have known positions). The number of Discrete scan lines per pass necessary depends on the uniqueness of the surface features. This method yields poor results in aligning the two datasets when scanning featureless surfaces which are relatively equidistant to the scanner throughout the scan pass as scans to either side of the Discrete scan lines will look very similar. Assuming that the surface either has unique features or sloped surfaces with respect to the scanner, the correct offset can be automatically optimized with good accuracy. The efficiency of data collection is less than the Encoder or Velocity methods since it requires collection of sparse Discrete data in addition to the Velocity data as well as an additional post-processing step to identify the correct offsets for each pass.

However, it is capable of collecting dense data much more efficiently than the Discrete method alone, and is not limited by the lack of encoder signal or consistent start timing.

For the PAMA machine at the Caterpillar Tech Center, a variety of tests were run to determine the suitability of each method. Figure 52 shows the result of a test in which a sloped part was scanned multiple times using the same scan path. The resulting measurements show that the PAMA machine has significant variability in the offset due to startup timing. This meant that the pure velocity method was not viable on the machine and so tests on the PAMA using in-machine scanning were done using the hybrid method – correcting the offset from the dense velocity method data with sparsely collected discrete data.

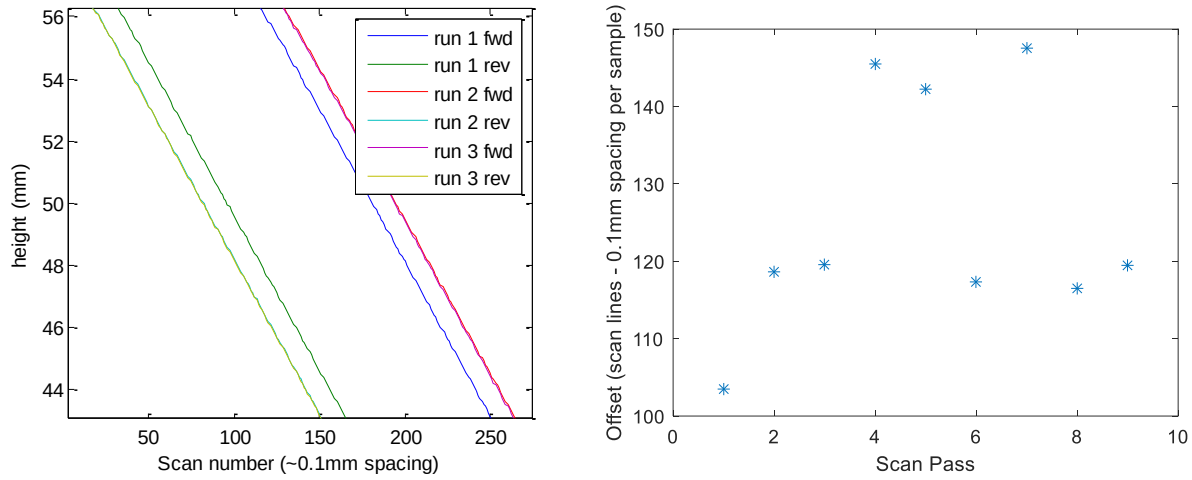


Figure 52: Velocity method PAMA offset consistency test (a) Raw data from same commanded portion of slope, measuring starting from each end. (b) Number of scan samples offset from true location for 9 iterations travelling in the same direction.

Figure 53 shows the result of a test to determine whether the dwell time of the machine at each sample location affected the accuracy of discrete method sampling. Sample points along a sloped surface were tested using discrete sampling and a variety of dwell times. The results all ended up being within a few microns of each other and did not have any discernable relationship to the dwell time, therefore it was concluded for the PAMA machine that simply stopping at each discrete point with no dwell was sufficient for full accuracy of discrete method sampling.

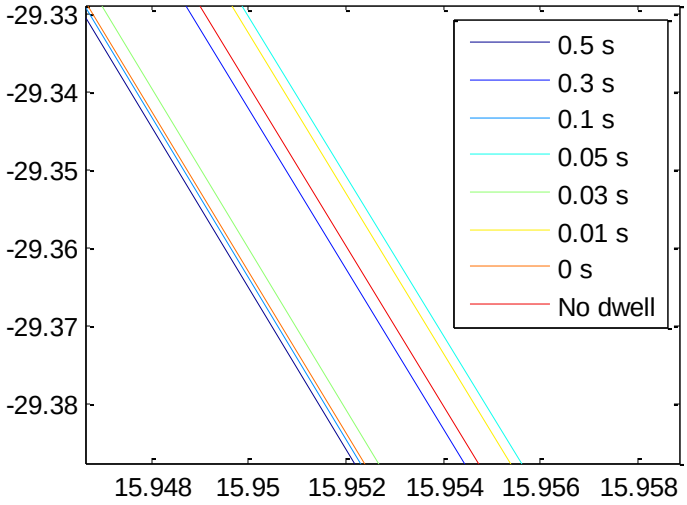


Figure 53: Discrete sampling dwell time test

The time required for the machine to reach a constant velocity was also tested. Figure 54 shows the result of measuring a flat sloped surface at three different speeds. The starting location was on the surface of the slope and the resulting measurements were aligned so that the start of visible motion was aligned with a time of 0.

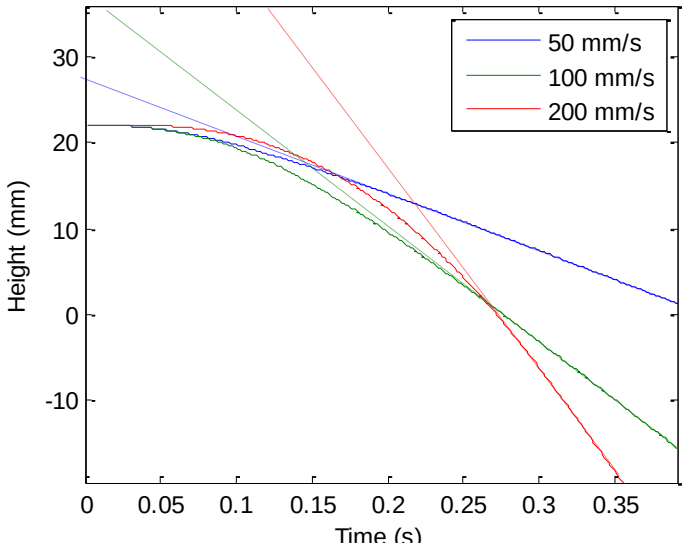


Figure 54: Acceleration profile testing

In order to accurately measure the location of a part using the laser scanner, the position and orientation of the scanner with respect to the machine's tool frame must also be identified. A calibration piece was developed which would allow the position and orientation of the scanner frame with respect to the machine end effector to be determined. The design drawing for the part is shown in Figure 55. The key aspect needed from the piece is a set of multiple 3D positions

relative to the machine base frame which can be uniquely identified from a variety of variations in scanning paths. To this end, a set of five flat surfaces were created at the top of the part which have enough slope to be clearly distinguishable, but also shallow enough slope so that all five surfaces can be scanned simultaneously from a single pass of the scanner. By fitting a plane to each of the five surfaces, the intersection points of the planes define four unique points which can be identified even if the exact spacing of the scanner samples does not capture the precise location of the corners.

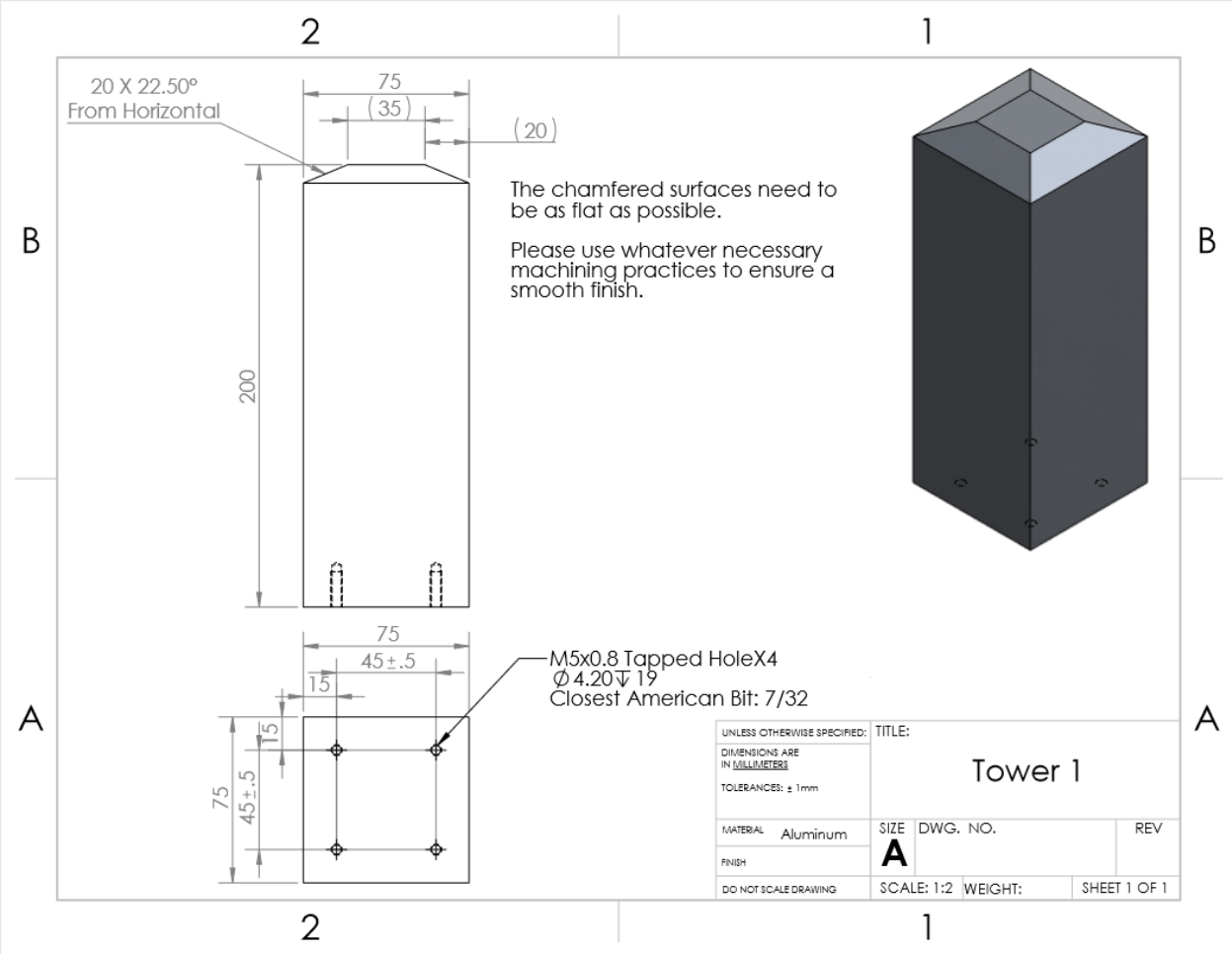


Figure 55: Calibration Piece

Figure 56 shows an example of the base frame (at the center of the table), tool frame, scanner frame and calibration feature.

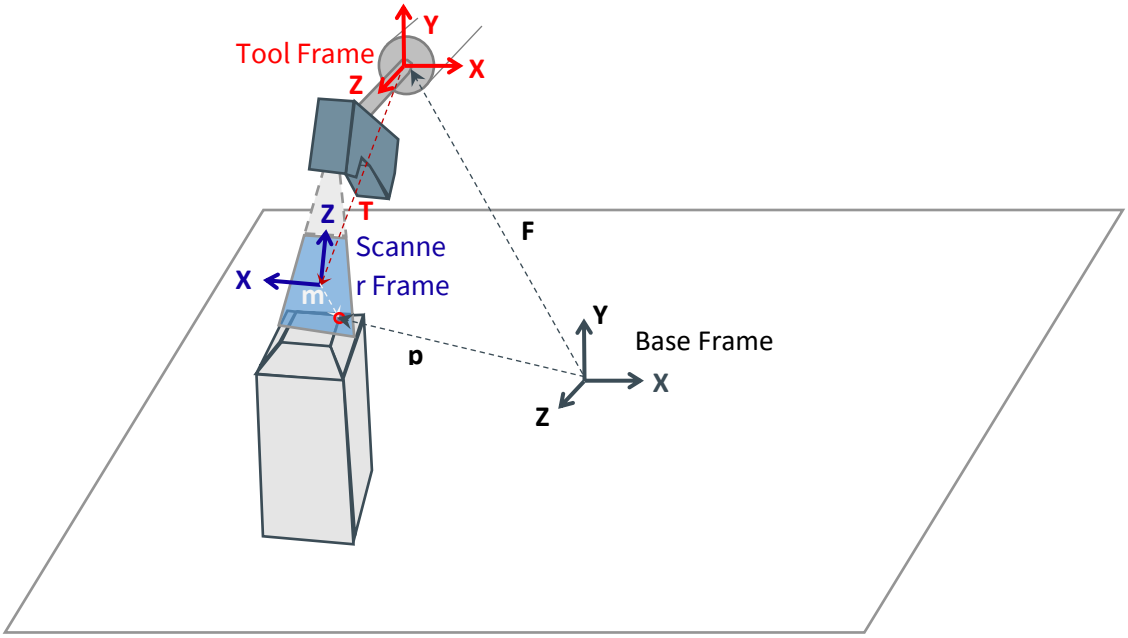


Figure 56 **Reference frames applicable to calibration.**

For any given point that is measured by the scanner, its position measured in the base frame is *p*, while the scanner records an x and z component in the scanner frame. The scanner measurement relates to the base frame position of the point via the 4x4 transformation matrices associated with the machine forward kinematics ***F*** (which are a function of the commanded x,y,z,w, and b axes positions as well as the spindle angle s) and tool-scanner transformation ***T***.

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = [F(x,y,z,w,b,s)][T] \begin{bmatrix} m_x \\ 0 \\ m_z \\ 1 \end{bmatrix} \tag{64}$$

The y coordinate of the measurement will always be zero since the laser only captures measurements in the 2-dimensional scan plane formed by the projection of the laser stripe. The tool-scanner transformation matrix can be represented by a rotation matrix and a translation vector.

$$T = \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix}, R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \mathbf{0} = [0\ 0\ 0] \tag{65}$$

The forward kinematics transformation matrix will also be invertible (representing the transform from tool frame to base frame) and the inverse will have the form

$$F^{-1}(x,y,z,w,b,s) = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{66}$$

Applying the inverse of the machine kinematics to both sides of the original equation yields

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_x \\ 0 \\ m_z \\ 1 \end{bmatrix} \quad (67)$$

Which can also be written

$$f_{11}p_x + f_{12}p_y + f_{13}p_z + f_{14} = r_{11}m_x + r_{13}m_z + t_x$$

$$f_{21}p_x + f_{22}p_y + f_{23}p_z + f_{24} = r_{21}m_x + r_{23}m_z + t_y$$

$$f_{31}p_x + f_{32}p_y + f_{33}p_z + f_{34} = r_{31}m_x + r_{33}m_z + t_z$$

At this point there are two methods possible to solve for the unknown parameters. If the location of given identifiable points are known via touch probe measurements, then the only unknowns are the parameters of the tool-scanner transformation. The equation can then be rewritten in matrix form isolating these unknowns (where the right hand side is only the meaningful vector component of the vector result of $[F^{-1}\boldsymbol{p}]$ with the concatenated 1 in the 4th component stripped away.

$$\begin{bmatrix} m_x & m_z & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & m_x & m_z & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & m_x & m_z & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{13} \\ r_{21} \\ r_{23} \\ r_{31} \\ r_{33} \\ t_x \\ t_y \\ t_z \end{bmatrix} = [F^{-1}\boldsymbol{p}]_{3x1} \quad (68)$$

If the above solution is viewed as being of the form Ax=b, then the matrices found for each feature measured can be stacked:

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \quad (69)$$

For each feature point measured, the measurements and associated machine kinematics result in 3 additional rows. Since there are 9 unknowns, then a minimum of 3 independent feature points must be measured in order to find a unique solution. If more feature points are measured (or the machine axes which are independent of whichever axis is used for scanning are varied) then there will be more equations than unknowns and the solution will be given by a least square fit. The solution vector can then be reassembled into the 4x4 transformation matrix. Note that the resulting solution does not directly yield the 2nd column of the transformation matrix. However, given the assumption that this represents a rotation matrix, the three columns of the

rotation matrix portion should form an orthonormal basis of $\mathbb{R}^3$, so the 2nd column can be found by taking the cross product of the 3rd and 1st columns. Because of measurement noise, it is also possible that the resulting solution for the 1st and 3rd columns are not perfectly independent or of perfect unit length. To find the best fit orthonormal set, the vectors can be made into unit vectors by dividing by their magnitude. Then the directions of the vectors can be made independent by subtracting half of the projection of itself onto the other vector.

$$C_1' = C_1 - \frac{1}{2}C_2(C_1 \cdot C_2), C_2' = C_2 - \frac{1}{2}C_1(C_1 \cdot C_2), \quad (70)$$

An exaggerated example is illustrated in Figure 57.

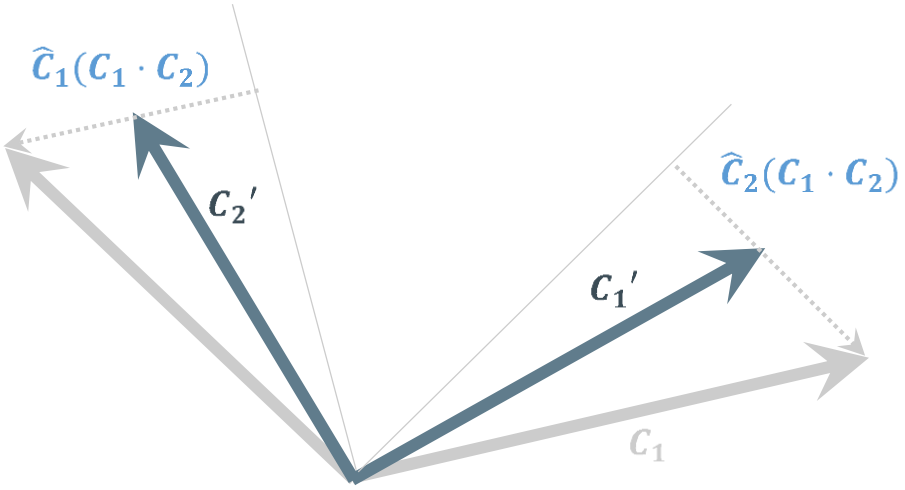


Figure 57 **Forming a best fit orthonormal basis.**

Alternatively, to using touch probe measurements, the locations of identifiable feature points can also be treated as unknowns themselves. In this case there are 3*N equations and 3*N+9 unknowns where there are N feature points. In this case the only way to add additional measurements is to vary the values of the forward kinematics. Though measuring the calibration part will nominally result in 3*N additional equations, many of these will not be independent. For the PAMA machine if the scan passes are run by moving the Y-axis, then measuring the feature points at an original configuration of the other axes, repeating the measurement with the X-axis shifted, and repeating again with the Z axis shifted is sufficient to identify the rotation matrix portion of the transformation but the translation vector will not be separable from the feature point coordinates. In order to separate the translation vector of the tool-scanner transform from the point coordinates, a measurement set must be taken with an alternate angle of the B-axis (establishes the location of the base frame y axis and therefore the x and z coordinates of the feature points) and another with the spindle at a different angle (establishes the tool frame z axis and therefore the y-component of the feature points after accounting for the y-position of the tool frame). The linear equation to be solved is then of the following form:

$$\begin{bmatrix} M_1 & {F_1^{-1}}_R & 0 & \cdots & 0 \\ M_2 & 0 & {F_2^{-1}}_R & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ M_N & 0 & 0 & \cdots & {F_N^{-1}}_R \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{13} \\ r_{21} \\ r_{23} \\ r_{31} \\ r_{33} \\ t_x \\ t_y \\ t_z \\ p_{1x} \\ p_{1y} \\ p_{1z} \\ p_{2x} \\ p_{2y} \\ p_{2z} \\ \vdots \\ p_{Nx} \\ p_{Ny} \\ p_{Nz} \end{bmatrix} = \begin{bmatrix} {F_1^{-1}}_t \\ {F_2^{-1}}_t \\ \vdots \\ {F_N^{-1}}_t \end{bmatrix} \quad (71)$$

Where the Ms represent matrices of the same form as the left hand side matrix in the previous method and corresponding to one scan pass. The F^{-1} matrices in the left side are the rotation matrix portion of the inverse of the machine forward kinematics at the instant that the measurement was taken (machine axis coordinates are known) and the F^{-1} matrices on the right hand side are those corresponding to the translation portions.

Figure 58 shows a comparison scan of a sample object using calibrations based on the two methods. Method 1, using the touch probe data was found to be much less accurate than Method 2, which relied solely on scanner measurements taken with different known machine variations in the rotary and linear axes during each scan pass of the calibration piece.

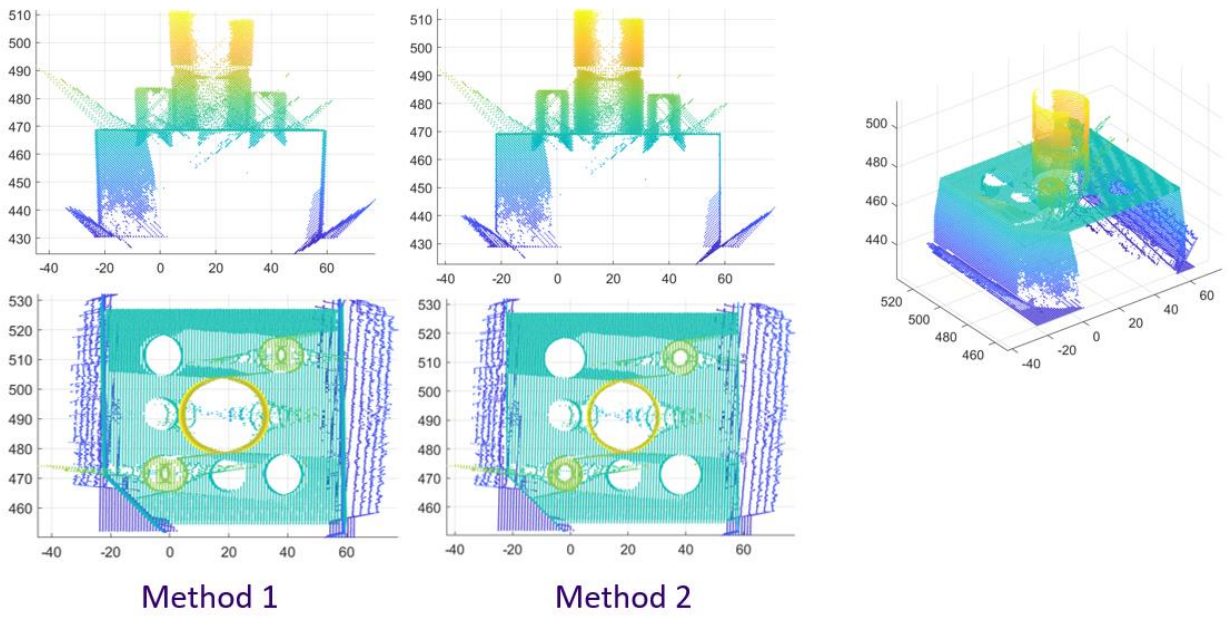


Figure 58: Comparison of calibration methods

The final calibration procedure used is given as follows:

Scanner Calibration

Steps 1 and 2 only need to be done once for a chosen calibration setup if the calibration piece is mounted in roughly the same location each time. Remaining calibration steps should be performed to identify the machine end effector to scanner head transformation each time the scanner is re-mounted.

1. Mount the calibration piece onto the machine table or fixturing in an area that the scanner can scan the angled top surfaces without collisions. For ease of path generation, it is usually best to align the piece with axis directions.
2. Generate a path of 5 scan passes (can add additional passes for increased accuracy of calibration result, but also requires increased time to complete the calibration) that include all 5 top surfaces of the calibration block in the scanning measurement range for each pass. All scan passes should be of the same length and use the Y-axis for scan pass motion (code and method could potentially be modified to allow for a different axis scan direction). Scan path locations should be chosen to fit the following format:
    a. Base pass – set the spindle angle to zero (or whatever base angle is needed to point the scanner generally down)
    b. Base pass shifted slightly (10mm) in X
    c. Base pass shifted slightly (10mm) in Z
    d. Base pass with B axis rotated slightly (10 deg) – also adjust X and Z as needed to ensure the scan pass still covers all 5 top surfaces
    e. Base pass with spindle rotated slightly (10 deg) – also adjust X and Z as needed to ensure the scan pass still covers all 5 top surfaces

3. Scan the calibration block using a continuous mode (also scan with discrete mode and perform data alignment if machine requires hybrid method due to an inconsistent delay in starting which prevents pure use of the velocity method.)
4. Open ProcessCalibScans.m and enter the starting positions of each scan pass used for the calibration scans as well as the file paths to the scan files and offset file.
5. Run ProcessCalibScans.m. Each scan pass will be displayed graphically and for each of the 5 surfaces on the calibration piece, a bounding polygon should be drawn which encompasses a portion of the flat surface, but does not include any of the edges. This is repeated for each of the scan passes. The program will then fit a plane to the data within each of the selected areas, calculate the intersection points between the planes and use the locations of these intersection points as well as the known differences in machine location for each scan pass to identify the transformation between machine end effector and scanner head. The result will be output to the MATLAB command line as well as saved as a variable in the MATLAB workspace. Outputs include the rotation matrix 'R' describing the orientation of the scanner frame with respect to the end effector frame, the translation 't' of the scanner frame with respect to the end effector frame, the full 4x4 transformation matrix 'transf', and the locations which were identified of the corner points of the calibration part in the machine base frame (this can be used as a sanity check to ensure that the calibration was successful if the location that the calibration part was mounted is known).
6. Copy the values of the generated transformation matrix 'transf' to the Vericut simulation by selecting the Scanner Tool from the machine model tree and then entering the values under the 'matrix' tab of the Configure Component panel.

Copy the values of the generated transformation matrix 'transf' to the MATLAB script used to generate the scanning path G-code and position files 'EditScanPath.m' and run the script on the scan path waypoint outline desired in order to generate position files with the correct scanner transformation. Alternatively, if position files have already been generated, the transformation matrix can be copied directly into each of the .csv files, replacing the previous transformation that had been used.

In order to plan and simulate scan paths without actual trial and error on the machine, a processing workflow using Vericut simulations was developed.

Vericut Setup

Vericut is used for generating the scan path and simulating the motion of the scanner to check for collisions and proper scan coverage. A model of the scanner head itself as well as a model representing the scanning beam measurement range were added to the model of the machine as shown in Figure 59.

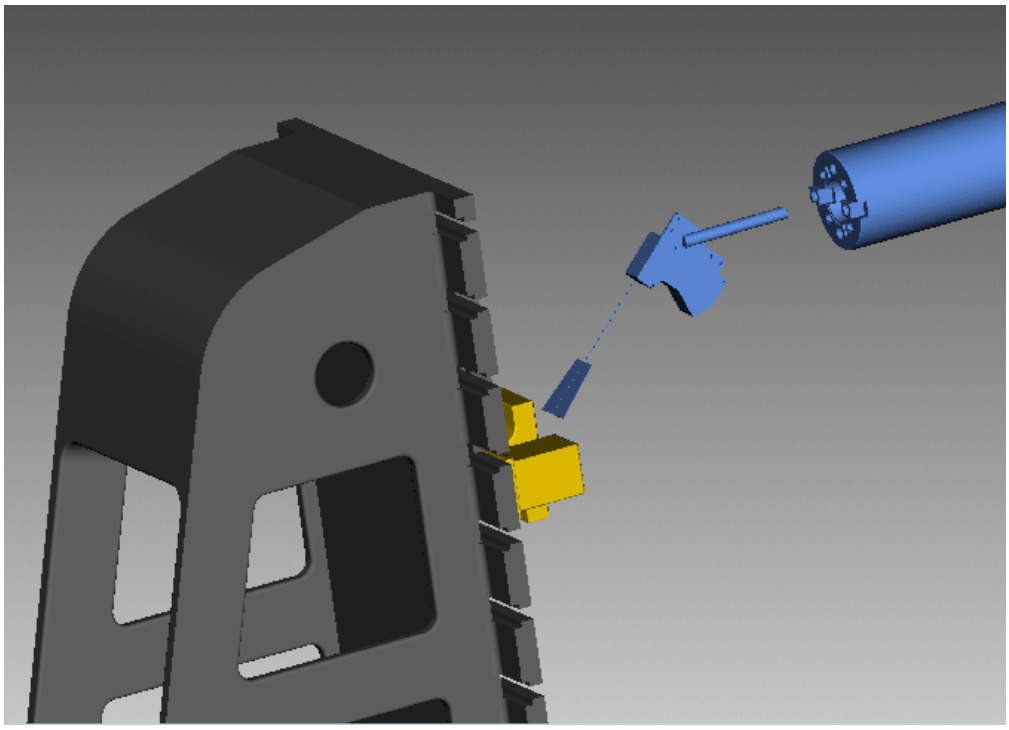


Figure 59: Simulation of In-machine Scanning with Vericut

To indicate when the scanner is collecting measurements, the visibility of the scanning beam measurement range model is linked with the M commands that are used to turn the measurement trigger on and off. In the experiments done on the PAMA, the coolant signal was repurposed to trigger the scanner turning on and off with the commands M98 and M9. The default within Vericut is for M8 and M9 to control the coolant signal while M98 calls a subroutine. Therefore, for the simulation, all program calls to M98 were re-routed to the M8 command by adding a substitute entry under the Advanced Control Options panel under Machine/Control. The M8 and M9 commands were also modified in the G-code processing panel (They are found under 'States') by right clicking on the 'Coolant On/Off' command, selecting 'Add/modify' and then selecting SetComponentVisibility, and setting the Value override to either 3 or 0 for turning on or off the visibility respectively, and the Text override value to the name of the model for the laser measurement range in the model tree.

Scan Path Generation

Vericut is used for generating the scan path by loading models of the machine tool, the scanner, the part to be scanned, and the mount for the part. The position of the mount and the part on the mount should be as close as possible to their real world locations to minimize the amount of manual correction. The position and orientation must be estimated prior to calibration, so it is best to mount the scanner aligned with the spindle angle of 0. After calibration the transformation identified can be updated in Vericut to improve the accuracy of the simulation. The steps for generating the scan path using Vericut are as follows:

1. Use the Manual Data Input mode (MDI) to jog the machine in Vericut such that the center of the scanner's measurement range is lined up with the leading edge of the desired feature and record the axis positions to the NC Block Entry list in the MDI window using 'Save Location to List'. Figure 60 shows an example waypoint list being created using the Vericut MDI.

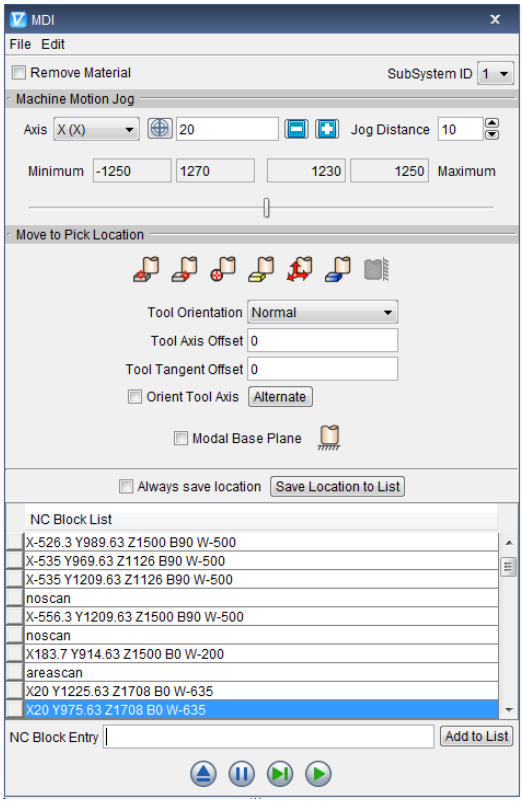


Figure 60: Waypoint List Creation in Vericut MDI

2. Jog the machine in Vericut such that the center of the scanner's measurement range is lined up with the trailing edge of the desired feature and record the axis positions to the NC Block Entry list. For each scan path segment, the machine motion should be limited to a single axis.
3. Repeat the process for each scan path segment.
4. If necessary, add waypoints to prevent collisions by entering a keyword such as 'noscan' in the text entry box and adding just before the waypoint with the 'Add to List' button.
5. For scanning large rectangular areas, instead of recording each start and end point directly, first add a keyword such as 'areascan' to the list to indicate the rectangular area input, then record three positions: the start and end point of the first pass across one end of the rectangular area, and the end point of the final pass (the adjustment between the two end points should be a change in a single axis perpendicular to the axis used for the scan pass segments themselves)
6. If the orientation of the scanner needs to be changed, add a spindle rotation command of the form SPOS=[angle], with a numeric value of the angle used in place of [angle].
7. Save the list to a text file from the MDI.

8. Open the EditScanPath.m matlab file and edit any parameters needed as detailed in the description of this function. Specify the file path to the text file generated in the MDI and the base filename for the output files.
9. Run EditScanPath.m. This will generate four files with names based on the output file base name and different endings as follows:
    a. [BaseName]GCode.MPF: File to be loaded on the machine to run continuous scanning (suitable for either encoder or velocity modes)
    b. [BaseName]DiscGCode.MPF: File to be loaded on the machine to run discrete scanning.
    c. [BaseName]PosFile.csv: File to be used by scanning acquisition code if scanning in a continuous mode.
    d. [BaseName]DiscPosFile.csv: File to be used by scanning acquisition code if scanning in discrete mode.
10. Either of the generated .MPF files can be loaded into the Vericut simulation by right clicking 'NCPrograms' in the project tree and selecting the file. This can then be run to test for any collisions after the path is edited with intermediate scan passes for intermediate scan passes for rectangular areas and any additional segment extensions for acceleration/deceleration.
11. The code should be verified on the actual machine with the speed turned down and the scanning head observed carefully for any potential collisions the first time it is run. If any corrections need to be made, use the MDI and re-open the saved .txt file to edit point axis commands as necessary, then repeat steps 7-11.

4.3.3.1 Formulation for a Single Virtual Gage

Let $\vec{r}_i$ be the i[th] point in the point set, *S*, that represents a planar surface on a part. After the rigid body transformation, T, is applied, the shortest distance from $\vec{r}_i$ to some plane, *A*, with equation, $ax + by + cz + d = 0$ is given by,

$$e_i = [a \quad b \quad c \quad d]T\begin{bmatrix}\vec{r}_i \\ 1\end{bmatrix}, \tag{72}$$

With small angle assumptions, a rigid body transformation can be written as,

$$T = \begin{bmatrix} 1 & -\alpha & \beta & \Delta x \\ \alpha & 1 & -\gamma & \Delta y \\ -\beta & \gamma & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{73}$$

where α, β and γ are the roll, pitch yaw angles in radian for angular motion and Δx, Δy and Δz describe linear motion. We define Ω as the set of all rigid body transformations where $-\delta_r \leq \alpha, \beta, \gamma \leq \delta_r$ and $-\delta_t \leq \Delta x, \Delta y, \Delta z \leq \delta_t$ (the angular rotations are limited by δ_r and the translations are limited by δ_t), shown in Figure 61.

A 3-D virtual gauge problem that attempts to make plane *A*, a support plane for *S* at a minimum distance of ε from it while minimizing the distance of the farthest point in *S* from it can be expressed as a constrained min-max linear programming problem,

$$\min_{T\in\Omega} q \quad such\ that\ 0 \ \leq \varepsilon \leq e_i \leq q, \forall\, r_i \in S \tag{74}$$

where q is the target function as well as an upper bound of all the distance of points r_i in *S* from *A*, and ε is the desired 'clearance from' or 'allowance for' the gauge to the point set as shown in Figure 62.

4.3.3.2 The multiple virtual gauge problem

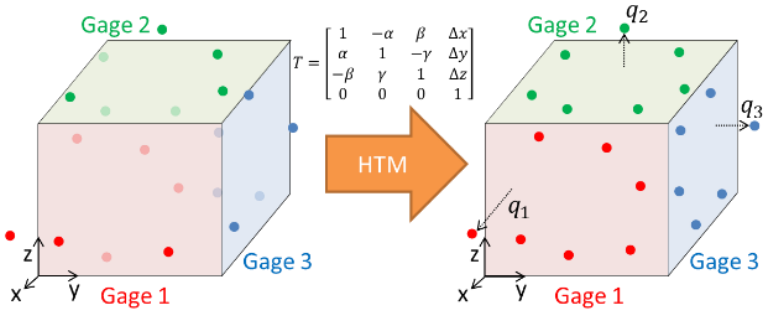


Figure 61 A single homogeneous transformation matrix rigidly displaces the point-sets so that their corresponding gage planes become supporting

For a part with multiple virtual gages specified, a single homogeneous transformation matrix, T, is used to simultaneously displace (rotate and translate) all the point sets so that their correspoding gage planes become supporting. This is shown in Figure 62.

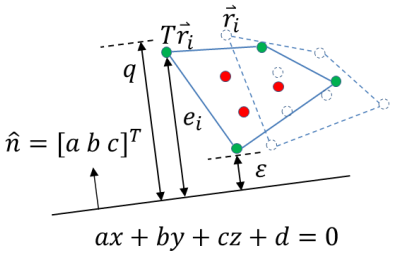


Figure 62 **The optimized rigid body transformation is found by minimizing the maximal distance.**

In an n-gauge problem we have n pairs of point sets and gage planes $\{S_i, p_i\}, i = 1, 2, \ldots n$. The distance $e_{i,j}$ of the j[th] point, $\vec{r}_{i,j}$ in i[th] point-set, S_i to the i[th] gauge plane, p_i is given by:

$$e_{i,j} = \vec{p}_i{}^T T \begin{bmatrix} \vec{r}_{i,j} \\ 1 \end{bmatrix} \leq q_i, \tag{75}$$

where q_i represents an upper bound on $e_{i,j}$, and $\vec{p}_i^{\,T} = [a_i \quad b_i \quad c_i \quad d_i]$ is the coefficient vector of the i[th] gauge plane, p_i with plane equation $a_i x + b_i y + c_i z + d_i = 0$, where $\|[a_i \quad b_i \quad c_i]^T\| = 1$.

The algorithm seeks to minimize the weighted sum of n distance upper bounds,

$$\min_{T\in\Omega} \sum_{i=1}^{n} f_i q_i \ \ such\ that\ 0 \le \varepsilon_i \le e_{i,j} \le q_i \ \forall\, r_{i,j} \in S_i, i = 1, \dots n \qquad (76)$$

where f_i, the weighting coefficient for the distance measure, q_i of the i[th] virtual gage has elements $f_j \ge 0, j = 1 \sim n$ and ε_i is the desired clearance specified for the i[th] gauge.

The linear program of Eq. (5) is bounded because Ω, the domain of its decision space, is bounded. If the linear programming problem is found to be infeasible, there are two possible reasons: a) the infeasibility is due to insufficient material on the part or b) Ω, the limitation on allowable transformations T is restricting the algorithm for obtaining a feasible solution. The reason for the infeasibility can be verified by examining the Lagrange multipliers of the constraints at termination. If any of the constraints that correspond to the limits imposed by Ω have non-zero values, then one can attribute the infeasibility due to restricting the set of feasible rigid-body transformation. The situation can be resolved either by relaxing Ω or by using a sequential programming approach (updating the problem linear program at the current value of T and restarting the optimization). In none of the constraints are associated with Ω, then the infeasibility is due to dimensional defects (e.g., insufficient material) to satisfy all the gages. In the context of the casting metrology problem described earlier, one can choose to reduce clearances or machining allowance requirements, relax constraints post by gages deemed less important than other, or reject the casting.

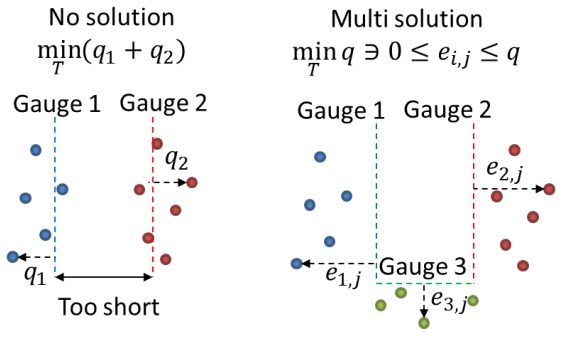


Figure 63 **(a) No solutions possible because of a dimensional defect; (b) In a multi-gage problem, when min-max objectives are used, feasible optimal solutions are found even when unbounded displacements (here in the vertical direction) are possible.**

4.3.3.3 Constructing locator frames for point-sets

As mentioned earlier, the virtual gage integrates the gage planes with sample data (point-cloud) from the physical part. The gage formulations are extracted from CAD models and it is necessary to create compatible reference frames between the point-cloud data and the CAD model. Shown in Figure 64 is an example of CAD model with its reference frame and location faces identified. In this version of the implementation of the software, we only use a 3-2-1 location scheme to identify three orthogonal planes that serve as primary, secondary and tertiary data planes.

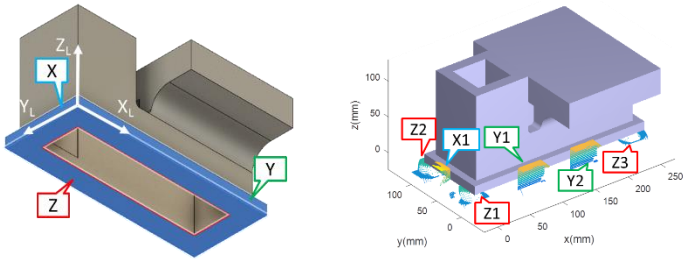


Figure 64 **Nominal locator frame with respect to the CAD model; (b) virtually located CAD model in point-cloud data scanned from the location surfaces of a fixture.**

First, point data that corresponds to location planes is extracted for the point-cloud. For example, if the point cloud is a fixture constructed with locator buttons, e.g., Figure 64(b), three separate location surfaces (Z_1, Z_2, Z_3,) provide data for fitting the primary datum or locating plane which, by convention, is selected as the XY plane of the reference frame that defines its z-direction. Likewise, the XZ-plane and YZ-plane are the secondary and tertiary datum reference, respectively. Each of these locator planes are obtained by identifying best-fit support planes for their corresponding point set. The planes are identified sequentially, with the primary (XY) plane being identified first, the secondary (XZ) plane next with the additional constraint that it is perpendicular to primary location plane, and finally the tertiary (YZ) plane is identified with constraints added to ensure that it is perpendicular to the other locator planes.

The primary datum plane, $a_z x_i + b_z y_i + c_z z_i + d_z$=0 is obtained by using the point sets associated with the location surfaces of the primary datum and finding the best supporting plane with the following non-linear optimization program:

$$\min_{\vec{p}_z, d_z} \max(\vec{p}_z{}^T \cdot \vec{r}_i + d_z)\ s.t.\ \vec{p}_z \cdot \vec{r}_i + d_z > 0;\ \|\vec{p}_z\| = 1\ \forall i, \qquad (77)$$

where $\vec{p}_z{}^T = [a_z \quad b_z \quad c_z]$ is the normal to the identified plane and d_z locates it in space and $\vec{r}_i$ represents a point in the point-sets associated with the primary location plane. The above optimization problem can be linearized and solved as a sequential linear program by preprocessing the data (For example, setting up the search for the optimal support plane is as a small perturbation on the best-fit least square plane for the given data). It should be noted that

this formulation can be reduced to the LP formulation given in Eq. (3). Additionally, because we are searching for a support plane, the point-set can be reduced (thus reducing the number of constraints), by only retaining the points on the convex hull of the point-set.

After the primary datum plane is fitted, the secondary datum plane can be obtained by exactly the same formulation given in Eq. (6) but with the addition of a constraint to enforce the perpendicularity requirement between the identified primary datum and the secondary datum. Thus, we have the following constrained optimization problem:

$$\min_{\vec{p}_y, d_y} max(\vec{p}_y{}^T \cdot \vec{r}_j + d_y)\ s.t.\ \vec{p}_y \cdot \vec{r}_j + d_y > 0;\ \ \|\vec{p}_y\| = 1;\ \vec{p}_y{}^T \cdot \vec{p}_z = 0\ \forall j \qquad (78)$$

where $\vec{p}_y{}^T = [a_y \quad b_y \quad c_y]$ is the normal to the identified secondary normal, and $\vec{r}_j$ represents a point in the point cloud from the surfaces associated with the secondary location plane.

The normal vector of the tertiary datum plane is fixed as it must be the cross product of $\vec{p}_y$ and $\vec{p}_z$, i.e, $\vec{p}_x = \vec{p}_y \times \vec{p}_z$. The optimization of the tertiary plan can then written as:

$$\min_{d_x} \max_k d_x\ s.t.\ \vec{p}_x \cdot \vec{r}_k + d_x > 0\ \forall k \qquad (79)$$

where $\vec{p}_x = \vec{p}_y \times \vec{p}_z, \vec{p}_x{}^T = [a_x \quad b_x \quad c_x]$ is the normal vector of the virtual X-plane and $\vec{r}_k$ represents a point in the point cloud of the X-locators.

After three datum planes are fitted, the locator frame can be constructed. The origin of the frame is given by solving three plane equations,

$$\begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} = -\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix}^{-1} \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \qquad (80)$$

A coordinate system can be represented by a 4 by 4 homogeneous transformation matrix (HTM). The locator frame identified above, in the coordinate systems of the point sets (scanner's coordinate system) is given by,

$$C_L^S = \begin{bmatrix} a_x & a_y & a_z & o_x \\ b_x & b_y & b_z & o_y \\ c_x & c_y & c_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (81)$$

Suppose, the solid model is constructed in a modeling frame, C_L^C the with the same primary, secondary and tertiary locator surfaces, then, located in the point-cloud data reference system, it is a 4 by 4 identity matrix. Brought into the point-cloud reference frame (scanner frame), it

would locate the part at its origin with its locator surfaces, aligned with the principal (XY, YZ, and ZX) planes (see Figure 65 (a)). The HTM, H which makes the point-cloud locator frame, C_L^S, identified by the aforementioned procedure coincident with the locator frame attached to the model is given by:

$$H \times C_L^S = C_L^C = I_4 \tag{81}$$

Thus,

$$H = (C_L^S)^{-1} \tag{82}$$

Therefore, HTM H bring the point cloud into alignment with the CAD model frame with the same datum planes. This is shown in Figure 65(b).

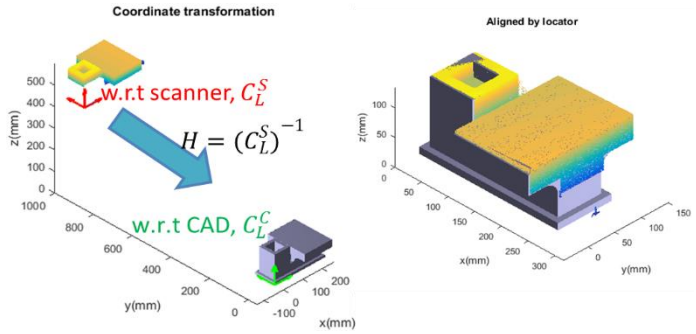


Figure 65 **(a). The attachment of a locator frame to the point set model. (b) The point cloud aligned with the part CAD model.**

4.3.3.4 Data sampling and filtering

After the point cloud is aligned with the part CAD model using the procedure discussed in the previous section, the point cloud can be segmented in to point-sets, such that each set is associated with a virtual gage. Each of these point-sets can then be processes to remove redundant points from the set.

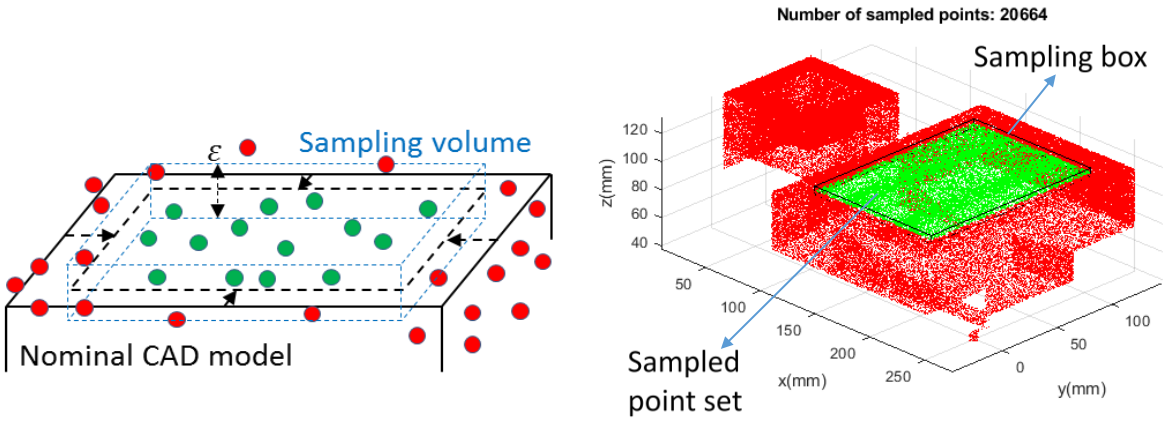


Figure 66 **(a) Schematic segmentation of point-cloud data using sampling volumes (b) An example of extracting a point set for a virtual gage from the part's point-cloud.**

Data segmentation: The extraction of points from the point-cloud to form a point-set for a virtual gage is accomplished by creating sampling volumes and classifying (deciding whether a point is in or out) the points against these volumes. These sampling volumes are constructed with the part CAD model (because the CAD environment has the appropriate tools to create and locate them relative to a face in part CAD model that will become a gage plane in the virtual gage). Besides identifying the points to be included in the point-set for a virtual gage, the sampling volumes are used for the removal of scanning artifacts (especially those produced near the edges of a surface during scanning). Figure 66(a) schematically depicts the use of sampling volumes to extract point-sets from the point cloud of the part and Figure 66(b) provides an example of the implementation of the sampling volume concept. In the current implementation, only rectangular boxes can be used as sampling volumes.

Point set thinning with convex hulls: The constrained optimization algorithms that implement the virtual gages are computationally intensive. Dense point sets generate a large number of constraints for a virtual gage, many of which are redundant. To reduce the computational time required to check a virtual gage, we reduce the number of constraints by thinning down the associated point-sets by identifying and eliminating redundant points.

Since our virtual gages essentially identify optimal support or classifying planes for point sets (i.e., planes that define half-spaces that either contain all the points or none), convex closures of the point-set play an important role in characterizing them relative to the gage planes. Therefore, only those points involved in the definition of a convex closure or hull (i.e., its vertices) need be considered. Other points, interior to the closure/ can be eliminated without fear of changing any metrics relative to the gage planes.

The convex hull of a finite point set, S is defined by the convex combination,

$$CH(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i x_i \,\middle|\, (\forall i: \alpha_i \geq 0) \wedge \sum_{i=1}^{|S|} \alpha_i = 1 \right\} \quad (83)$$

where x_i is the i[th] point in S.

A finite point set has unique convex hull, whose vertices, S' are the minimal possible subset of S that share the same convex hull. Thus, we have

$$CH(S) = CH(S') \quad (84)$$

All extremal (minimum or maximum) distances between the S and a support or classifying plane are defined by points in S'

Let S be a finite point set in 3-D space, and S' be its subset consisting all vertices of $CH(S)$. Assume there are $|S'|$ vertices in $CH(S)$, which is also the number of points in S'. Any point, $u \in S \notin S'$, i.e., interior to $CH(S)$ can be expressed by the convex combination of all points in S',

$$u = \left\{ \sum_{i=1}^{|S'|} \beta_i y_i \,\middle|\, (\forall i: \beta_i \geq 0) \wedge \sum_{i=1}^{|S'|} \beta_i = 1 \right\}, \quad (85)$$

where $y_i \in S'\ \forall i$.

If all points in S' lie on the same side of a given plane with plane equation, $ax + by + cz + d = 0$, we have,

$$\forall y_i \in S': {y_i}^T \cdot \hat{n} + d > 0, \quad (86)$$

where $\hat{n} = [a \quad b \quad c]^T$ is the normal vector of the plane.

Substitute u into the plane equation to check if any given point u inside of $CH(S)$ locates on the same side of the plane,

$$u^T \cdot \hat{n} + d = \left\{ \sum_{i=1}^{|S'|} \beta_i ({y_i}^T \cdot \hat{n} + d) \,\middle|\, (\forall i: \beta_i \geq 0) \wedge \sum_{i=1}^{|S'|} \beta_i = 1 \right\} \quad (87)$$

According to equation 87, ${y_i}^T \cdot \hat{n} + d > 0$ and since by definition all β_i are positive, $u^T \cdot \hat{n} + d$ imust also also positive. Therefore, u and all vertices are on the same side, or, if the vertices of the convex hull are on the same side of a plane, every point inside the convex hull is also on the same side.

Also, the farthest and nearest points in a finite point set, S, from a support or classifying plane must be a vertex of its convex hull, $CH(S)$. The proof is given below.

A point inside of the convex hull can be expressed by the convex combination of the vertices,

$$u = \left\{ \sum_{i=1}^{|S'|} \beta_i y_i \,\middle|\, (\forall i: \beta_i \geq 0) \wedge \sum_{i=1}^{|S'|} \beta_i = 1 \right\}, \tag{88}$$

where $y_i \in S'\ \forall i$.

The distance from the point to a given plane, $ax + by + cz + d = 0$ can be written as,

$$u^T \cdot \hat{n} + d = \left\{ \sum_{i=1}^{|S'|} \beta_i ({y_i}^T \cdot \hat{n} + d) \,\middle|\, (\forall i: \beta_i \geq 0) \wedge \sum_{i=1}^{|S'|} \beta_i = 1 \right\} > 0 \tag{89}$$

where $\hat{n} = [a \quad b \quad c]^T$ is the normal vector of the plane.

Let y_m and y_n be the farthest and nearest vertices from the given plane. The upper bound of the distance is given by

$$\begin{aligned} u^T \cdot \hat{n} + d \leq & \left\{ \sum_{i=1}^{|S'|} \beta_i ({y_m}^T \cdot \hat{n} + d) \,\middle|\, (\forall i: \beta_i \geq 0) \wedge \sum_{i=1}^{|S'|} \beta_i = 1 \right\} \\ & = {y_m}^T \cdot \hat{n} + d \end{aligned} \tag{90}$$

Similarly, the lower bound of the distance can be obtained. We have,

$$0 < {y_n}^T \cdot \hat{n} + d \leq u^T \cdot \hat{n} + d \leq {y_m}^T \cdot \hat{n} + d \tag{91}$$

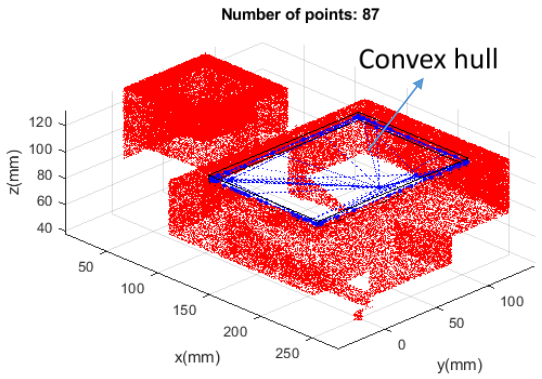


Figure 67 **A point set of about 20K points, filtered to 87 points using a convex hull filter.**

Thus, when we are working with gage planes and point sets for our virtual gages, only the vertices of the convex hull need be considered in the virtual gauge algorithms. In general, we expect $|S'| \ll |S|$, and the computing of $CH(S)$ is computationally much less intensive than solving the constrained optimization problem of the virtual gages. Thus, we expect to greatly reduce the

computational effort by only using the vertices of convex hulls of the point-sets instead of the entire point set for the implementation of the virtual gage algorithm. Figure 67 shows the use of a convex hull filter in replacing the point-set extracted by a sampling volume.

Then, two methods have been developed to achieve compensated tool path based on the HTM that generated from Virtual gage. First approach, the software read the existing program and only change the work offset; with second approach, the NC program will be modified to achieve the adaptive tool path.

Automated work offset modification

The system was demonstrated on a 3+1 axis CNC machine set-up with rotating table. A simplified schematic of the machine model with axes definitions is shown in Figure 68. In such systems, each face of table rotation typically comprises of a unique work offsets saved in different G codes (G5X series such as G55, G56, etc. or G5XX series such as G506, G507, etc. depending on the type of the machine). The new cutting tool positions are transformed from $X_1\ Y_1\ Z_1$ to $X_2\ Y_2\ Z_2$ for a table rotation of θ using a standard Y rotation matrix as given by the following equation:

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} * \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \quad (92)$$

For a complex part with multiple faces that require machining in different radial directions, it is evident that different set of offset values are required for different faces as the rotation about Y changes the cutting tool position in Z and X directions. To implement this, the offset values calculated using virtual gage block analysis are first transformed into the machine space using the following equation:

$$\begin{bmatrix} X_{cor} \\ Y_{cor} \\ Z_{cor} \end{bmatrix} = \begin{bmatrix} \cos(b_{tab}) & 0 & -\sin(b_{tab}) \\ 0 & 1 & 0 \\ \sin(b_{tab}) & 0 & \cos(b_{tab}) \end{bmatrix} * \begin{bmatrix} x_{cor} \\ y_{cor} \\ z_{cor} \end{bmatrix} \quad (93)$$

b_{tab} is the table rotation value for the offset being compensated and b_{tab} rotation matrix maps the linear translations from virtual gage block to the corrected table values as $X_{cor}\ Y_{cor}\ Z_{cor}$. The virtual gage block values are entered in $x_{cor}, y_{cor}, z_{cor}$. The corrected table values are then added to the existing work offsets for the part, if any, and the values are then mapped to the table values with another transformation using the standard Y rotation matrix with the b_{cor} as θ to account for the Y rotation from virtual gage block analysis. The final offset compensation equations for the 3+1 CNC axis machine setup are given below:

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{bmatrix} = \begin{bmatrix} \cos(b_{cor}) & 0 & -\sin(b_{cor}) \\ 0 & 1 & 0 \\ \sin(b_{cor}) & 0 & \cos(b_{cor}) \end{bmatrix} * \left(\begin{bmatrix} X_{cor} \\ Y_{cor} \\ Z_{cor} \end{bmatrix} + \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \end{bmatrix} \right) \quad (94)$$

$$B_{new} = b_{cor} + B_{old}$$

X_{new}, Y_{new}, Z_{new}, and B_{new} are the compensated offsets, and X_{old}, Y_{old}, Z_{old}, and B_{old} are the original offsets.

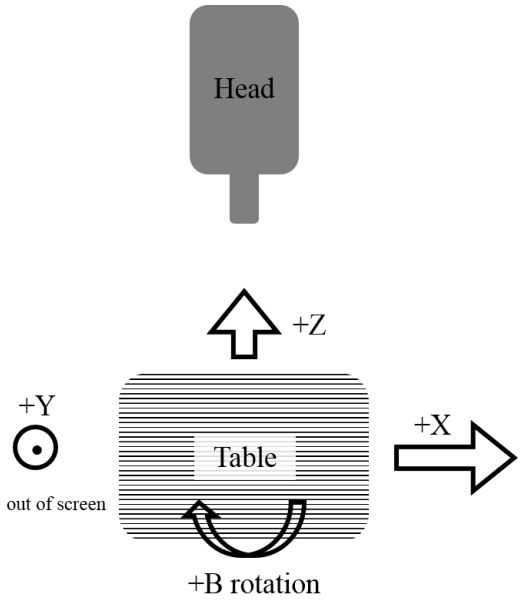


Figure 68 Simplified schematic showing different axes on the machine.

The program is written in Matlab. When compiled, the program prompts the user to input values from the virtual gage block including x_{cor}, y_{cor}, z_{cor} and b_{cor} values and the existing NC program. The program then stores the values in corresponding matrices and searches for any existing work offsets starting with G5X…X series used in the program and its line number. A snippet of the program performing the above two functions is shown below:

```
%--------------------------------------------------------------------------
%Dialog Box Input

prompt = {'Enter X Value:','Enter Y Value:','Enter Z Value:','Enter Y Rotation Value (deg):'};
title = '4DOF HTM Values';
response = inputdlg(prompt,title);
response = str2double(response);
if isempty(response) || any(isnan(response))
    fprintf('Invalid input for 4DOF HTM Values.\nExiting Program...\n')
    return
end

%--------------------------------------------------------------------------
%Rotations and translations are based on program zero origin with B0.

trans = response(1:3); %linear translations
rot = [0;response(4);0]; %xyz rotations in degrees (nothing can be done with x or z)

%--------------------------------------------------------------------------
%Getting info to open original G-code file
[openName, openPath, ~] = uigetfile({'*.mpf','G-Code Files (*.mpf)';...
                                     '*.txt','Text Files (*.txt)';...
                                     '*.*', 'All Files (*.*)'},...
                                    'Select the G-Code file to compensate');
if openName == 0
    fprintf('No file was selected for input\nExiting program...\n');
    fclose('all');
    return
end
%Extracting original file's extension
[~, openExt] = strtok(openName,'.');
%Opening file as read only
fin = fopen(strcat(openPath, openName),'r');

%Iterating through lines of original g-code file and placing in cell matrix
kk=1;
tline = fgetl(fin);
A{kk} = tline;
while ischar(tline)
    kk = kk+1;
    tline = fgetl(fin);
    A{kk} = tline;
end
fclose(fin);

%--------------------------------------------------------------------------
%Getting Ready for compensation

%Search for which workoffset are used and their line numbers
fndcount = 0;
fndoffset = cell(1,2);
for jj=1:length(A)-1
    [startI, match] = regexp(A{jj},...
        '(?:\<G5[4-7]\>)|(?:\<G5\d(?:(?<=0)[5-9]|(?<!0)\d)\>)',...
        'start','match');
```

New frame variables are defined in the program to be added in the additional lines of the modified NC program. Therefore, the program checks for any conflicting names in using variables names in which case the variable names should be changed. For example, UOLDWO represents saving the old work offset temporarily during the program implementation. A snippet from the program that introduces the new variables and scans the entire NC program for repetitions and conflicting variables is shown below:

```
%Verify that new frame variables and temporary offset variables do not have
%conflicting names
var6char = ['UOLDWO';'TABMAT';'CORMAT';'TABWOX';'TABWOY';'TABWOZ';...
    'CORWOX';'CORWOY';'CORWOZ'];
var4char = ['REGI';'BTAB';'XCOR';'YCOR';'ZCOR';'BCOR'];
var6char = strcat(var6char,'%c');
var4char = strcat(var4char,'%c');


subscript = 64;
noexitwhile = true;
while noexitwhile
    jj = 1;
    subscript = subscript+1;
    noexitinner = true; %Boolean variable to control loop
    while noexitinner && jj<length(A)-1
        for jjj = 1:size(var4char,1)
            if ~isempty(strfind(A{jj},sprintf(var4char(jjj,:),subscript)))
                for jjjj = 1:size(var6char,1)
                    if ~isempty(strfind(A{jj},sprintf(var6char(jjj,:),subscript)))
                        noexitinner = false;
                    end
                end
            end
        end
        jj = jj+1;
    end
    if noexitinner
        noexitwhile = false;
    end
    %Error checking
    if subscript > 90
        fprintf('\nThe program could not choose a valid variable ');
        fprintf('name for offset calculations.\nExiting program...\n\n');
        fclose('all');
        return
    end
end
```

The definition of temporary variables and saved offsets provides the basis for next implementation of the insertion of the offset calculation matrix equations in the modified NC program. The principal concept of this software is not modify the existing lines of G-code in the program but introduce easily distinguishable additional lines in the program for the calculations to occur in the NC machine controller. Therefore, the required variables should be defined in the modified NC program before the machining commands are run on the controller. Therefore, the frames are initialized and defined as follows:

```
%--------------------------------------------------------------------------
%Creating pre-initialization, initialization, and restore block to insert
%into G-Code

preini_block = cell(length(uniqueoffsets)+6,1);
ini_block = cell(length(uniqueoffsets),1);
restore_block = cell(length(uniqueoffsets),1);
preini_block{1} = sprintf('DEF INT %s',var4char(1,:));
for jj=1:length(uniqueoffsets)
    if length(uniqueoffsets{jj})==3
        temp = uniqueoffsets{jj};
        register = temp(end)-3;
    elseif length(uniqueoffsets{jj})==4
        temp = uniqueoffsets{jj};
        register = temp(end-1:end);
    else
        fprintf('\nError parsing unique offsets from file\nExiting ');
        fprintf('program...\n');
        fclose('all');
        return
    end
    preini_block{jj+1} = sprintf('DEF FRAME %s%c',var6char(1,:),jj+64);
    ini_block{jj} = sprintf('%s%c=$P_UIFR[%s]',var6char(1,:),jj+64,register);
    restore_block{jj} = sprintf('$P_UIFR[%s]=%s%c',register,var6char(1,:),jj+64);
end
preini_block{length(uniqueoffsets)+2} = sprintf('DEF REAL %s[3,3]',var6char(2,:));
preini_block{length(uniqueoffsets)+3} = sprintf('DEF REAL %s[3,3]',var6char(3,:));
preini_block{length(uniqueoffsets)+4} = sprintf('DEF REAL %1$s, %2$s, %3$s, %4$s, %5$s',...
    var4char(2,:),var4char(3,:),var4char(4,:),var4char(5,:),var4char(6,:));
preini_block{length(uniqueoffsets)+5} = sprintf('DEF REAL %1$s, %2$s, %3$s, %4$s, %5$s, %6$s',...
    var6char(4,:),var6char(5,:),var6char(6,:),var6char(7,:),var6char(8,:),var6char(9,:));
preini_block{length(uniqueoffsets)+6} = 'STOPRE';
```

The program then prints additional lines to implement offset calculations in controller to the existing NC program and then saves the new .MPF file in a user specified location with a new name. The snippet of the program with this implementation is shown below:

```
%Compensation block
comp_block = cell(17*length(uniqueoffsets)+7,1);

cb_const = {...
    sprintf('%1$s[0,0]=SET(COS(-%2$s),0,SIN(-%2$s),0,1,0,-SIN(-%2$s),0,COS(-%2$s))',var6char(2,:),var4char(2,:));
    'STOPRE';
    sprintf('%1$s=((%3$s*%2$s[0,0]+%4$s*%2$s[0,2])+($P_UIFR[%5$s,X,TR]+$P_UIFR[%5$s,X,FI]))',var6char(4,:),var6char(
    sprintf('%1$s=%2$s+($P_UIFR[%3$s,Y,TR]+$P_UIFR[%3$s,Y,FI])',var6char(5,:),var4char(4,:),var4char(1,:));
    sprintf('%1$s=((%3$s*%2$s[2,0]+%4$s*%2$s[2,2])+($P_UIFR[%5$s,Z,TR]+$P_UIFR[%5$s,Z,FI]))',var6char(6,:),var6char(
    'STOPRE';
    sprintf('%1$s=%2$s[0,0]*%3$s+%2$s[0,2]*%4$s',var6char(7,:),var6char(3,:),var6char(4,:),var6char(6,:));
    sprintf('%1$s=%2$s',var6char(8,:),var6char(5,:));
    sprintf('%1$s=%2$s[2,0]*%3$s+%2$s[2,2]*%4$s',var6char(9,:),var6char(3,:),var6char(4,:),var6char(6,:));
    'STOPRE';
    sprintf('$P_UIFR[%1$s]=CTRANS(X,%2$s,Y,%3$s,Z,%4$s,B,(%5$s+$P_UIFR[%1$s,B,TR]+$P_UIFR[%1$s,B,FI]))',var4char(1,:
    'STOPRE';
    ''};
```

```
%--------------------------------------------------------------------------
%Creating new G-Code file in user specified location

savePath = uigetdir(openPath,'Select Folder to Save Updated G-Code in');
if savePath == 0
    fprintf('\nNo folder was selected for output\nExiting program...\n');
    fclose('all');
    return
end
saveName = strrep(openName, openExt, strcat('_NEW', openExt));
fout = fopen(strcat(savePath, '\', saveName),'w');
```

The example program shown in above utilizes the variables that are specific to the Siemens Sinumerik controllers. The program can be modified by changing the variable names to correspond to controllers made by different manufacturers.

Regeneration of NC Program using Siemen's NX Journal

NX journaling is a tool used by engineers to automate certain procedures. A journal is similar to a macro but involves programming, usually written with Visual Basic (VB). However, NX journaling is not limited to VB as the programming language. Programming languages such as C#, C++, Java, or Python could also be utilized but VB is the most commonly used program. A typical journal creation is very similar to creation of a "macro" in NX software. A macro option in NX allows recording a series of operations performed in the CAD space on the file to a file that can be saved. Typically, the user has to begin recording the journal and then perform the desired procedures. After the journal has been first created, it can be modified to fit any desired function or model. Journals are also run similar to macros. They must be run by the user and be supplied with any requested inputs as programmed.

In this project, a journal was created in order to transform the tool paths of a CAD file created in a manufacturing or CAM module of Siemens NX software. The method assumes that the user utilizes NX software to create tool paths and generates the final NC program using appropriate post processor files. The journal transformation is a rigid body translation and rotation of the defined tool paths based on the origin of the machine co-ordinate system defined in the CAD model. The transformation utilizes the TransformObject() function available in NX software. After transformation, the journal also performs the post processing of the transformed tool paths based on the post files selected by the user. Therefore, this method assumes that the user has all required post files to obtain the final corrected NC program. This method is highly adoptable to any multi-axes CNC machine set-up if the tool paths are created and post processed using Siemens NX software.

In this method, the journal program is created using Visual Basic programming language by utilizing the pre-defined functions and syntax that is compatible with Siemens NX software. The

program first gathers the responses of the user to save the transformation values including X, Y, Z, translations and X, Y, Z rotations. A typical programming syntax for collecting the X translation from user reads as below:

```
'X translation
Do
    xTrans = NXInputBox.GetInputString("Enter X translation", "X Translation", " ")
    'if cancel is pressed, exit sub
    If xTrans = "" Then Exit Sub
Loop Until Double.TryParse(xTrans, useless)
```

The program then collects all the defined toolpaths or “NX Program Operations”. After the collection of all operations in one array, toolpath transformations are performed using the defined function provided for NX programming applications called transformBuilder.Commit(). The toolpaths are then also post processed using the journal.

```
Dim objectsToTransform() As CAM.CAMObject = _
    workPart.CAMSetup.CAMOperationCollection.ToArray()

Dim transformBuilder As CAM.OperationTransformBuilder
transformBuilder = _
    workPart.CAMSetup.CreateOperationTransformBuilder(objectsToTransform)

Dim Xtranslate As Scalar
Xtranslate = workPart.Scalars.CreateScalarExpression(expXtranslate, _
    Scalar.DimensionalityType.Length, SmartObject.UpdateOption.AfterModeling)

'Doing the transformation
transformBuilder.Commit()
transformBuilder.Destroy()

'Postprocessing the transformed toolpaths

Dim postUndo As Session.UndoMarkId
postUndo = theSession.SetUndoMark(Session.MarkVisibility.Visible, _
    "PostProcess")

Dim ncPath As String
Dim ncName As String
Dim iniText As String = _
    IO.Path.GetFileNameWithoutExtension(workPart.FullPath) & ".mpf"
```

4.3.4 Users & Use Cases

The in-machine scanning process was tested using the PAMA machine at the Caterpillar Tech Center to scan a small test casting. The 3D point cloud data generated was then utilized by the virtual gage software. The part was artificially perturbed by a known amount from the nominal mounting location and the generated 3D point cloud was used to identify the offset in position and orientation of the part with respect to the nominal mounting location and modifications to the machining code were generated using the adaptive machining software components.

The scan data of the raw casting is given in the scanner coordinate system, which is different from the solid CAD model's reference coordinate. To make two coordinate systems coincide with each other, six locators, which are fixed on the tombstone to clamp the casting, are scanned, and a locator frame can be fitted. Three Z-locators are placed between the tombstone and the casting; two Y-locators hold the casting from the beneath, and the X-locator clamps the casting on the left. The origin of the fitted locator frame locates at the corner of the solid model.

The setup of the clamps and six locators is shown in Figure 69. In order to examine if the algorithm can capture linear displacement errors along X and Y-axes, spacers are placed between locator X1 and the casting as well as between the Y-locators and the casting to simulate the translational errors. Similarly, a small angular error about the Y-axis and linear error about Z-direction are generated by the spacers between locator Z1, Z2 and the casting.

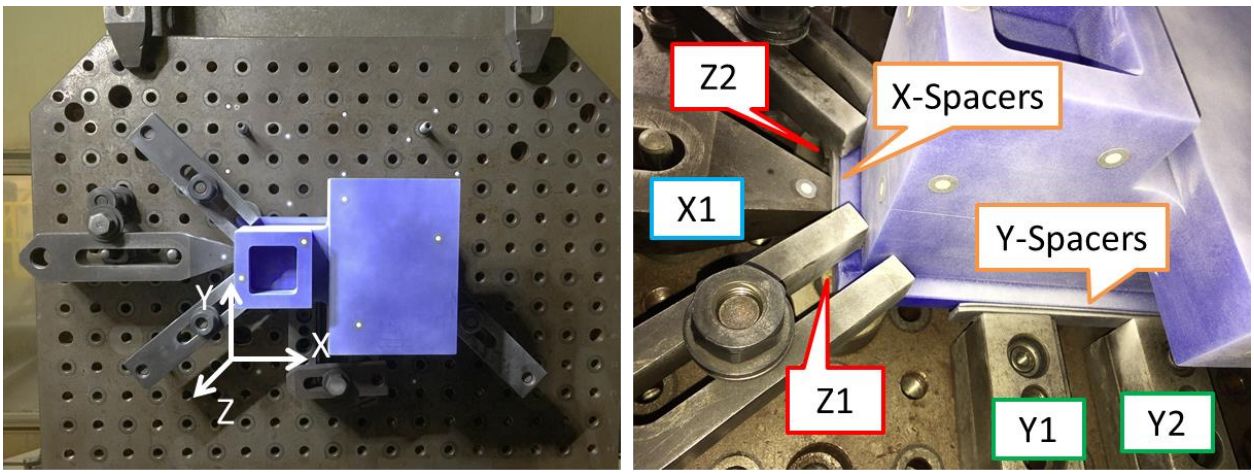


Figure 69 **(a) The setup of the wax casting clamped on the machine table; (b) Spacers placed between locators and the casting.**

Program 1 extracts 13 point sets as shown in Figure 70(a) from the raw point-cloud data (see Figure 70(b)). Seven of the 13 surfaces including the top and side of the flange, two inner walls and two outer walls and the top of the tower(see Figure 70(b)) are checked against eight virtual gauge parameter sets listed in Table 1, which are obtained from the dimensional and the tolerance requirements of the final casting in Figure 71.

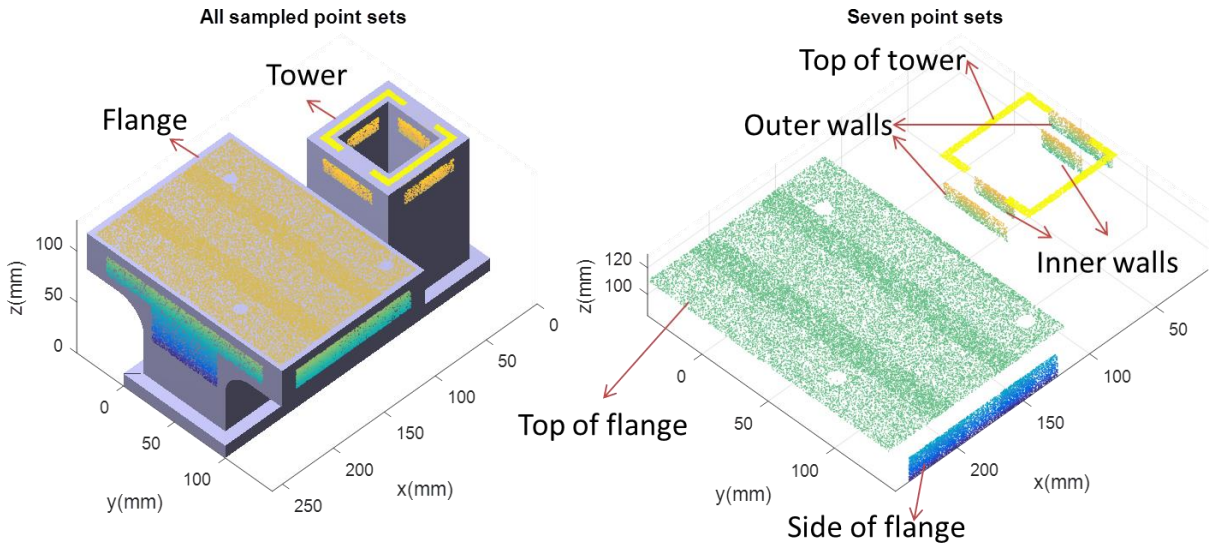


Figure 70 (a) **13 point sets extracted from the original data set; (b) 7 point sets used as constraints in the optimization procedure.**

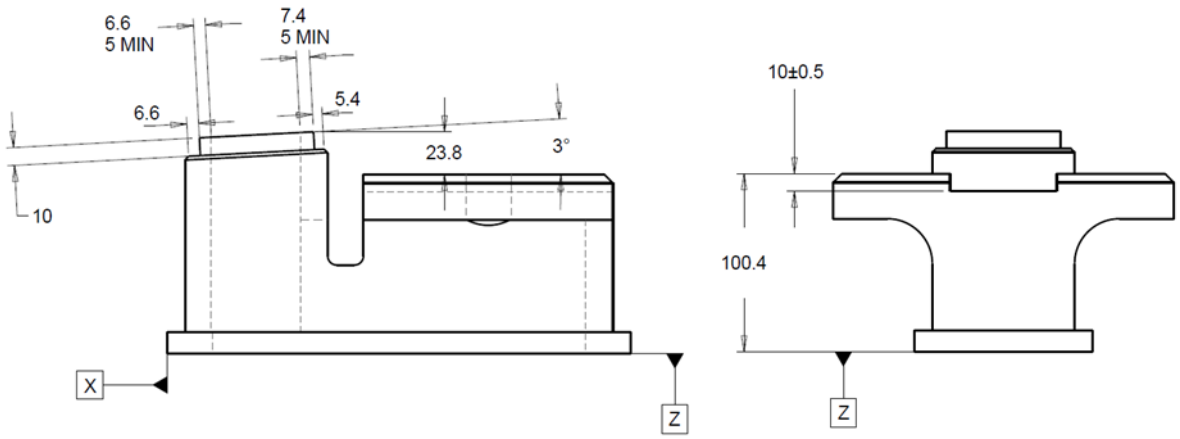


Figure 71 **GD&T requirements of the scaled part (side view); (b) Front view**

Table 19 **Virtual gage parameter sets**

	a	b	c	d
Top of flange	0	0	1	-109.30
Top of flange	0	0	-1	111.50
Side of flange	0	1	0	-145.80
Outer wall 1	1	0	0	-87.43

Outer wall 2	-1	0	0	12.42
Inner wall 1	-1	0	0	75.62
Inner wall 2	1	0	0	-23.48
Top of tower	0	0	1	-124.20

The machine used in the test has three linear axes and one rotary axis, which only allows the table to rotate about Y-axis. Program 2 establishes a linear programming problem given by equation 95 and solves the optimal rigid body transformation with four degrees of freedom,

$$T = \begin{bmatrix} 1 & 0 & \beta & \Delta x \\ 0 & 1 & 0 & \Delta y \\ -\beta & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Where β is restricted by the small angular assumption, $-0.05 \leq \beta \leq 0.05$.

The linear programming problem given by equation 95 can be written by,

$$\min_{T \in \Omega} \sum_{i=1}^{8} q_i \ such\ that\ 0 \leq e_{i,j} \leq q_i, \quad (96)$$

where $e_{i,j}$ and q_i are defined in equation 96

Program 2 returns a unique, optimal HTM for the 4-axis machine,

$$T^* = \begin{bmatrix} 1 & 0 & -0.0267 & -3.12 \\ 0 & 1 & 0 & -3.09 \\ 0.0267 & 0 & 1 & -7.19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (97)$$

In the last part of program 2, the effect of the optimal HTM is reviewed and visualized. T^* rotates the casting by -0.0267 radian (-1.298 degrees) along Y-axis, which is close to the angular offset, generated by the spacers placed between spacer Z1, Z2 and the casting. The point set, which represents the top of the flange, is tilted, but the effect of the rigid body transformation rotates the point set back to the horizontal position, shown in Figure 72 and Figure 73. The whole point set is translated -3.1171 and -3.0908 mm along X and Y direction, which are close to 3.00 mm, the thickness of the spacers placed between the casting part and the Y-spacers. The top of the flange is squeezed by two gauges, $z = 110.4 \pm \frac{\epsilon}{2}$ mm, where ϵ is the minimal possible thickness of the point set along Z-axis.

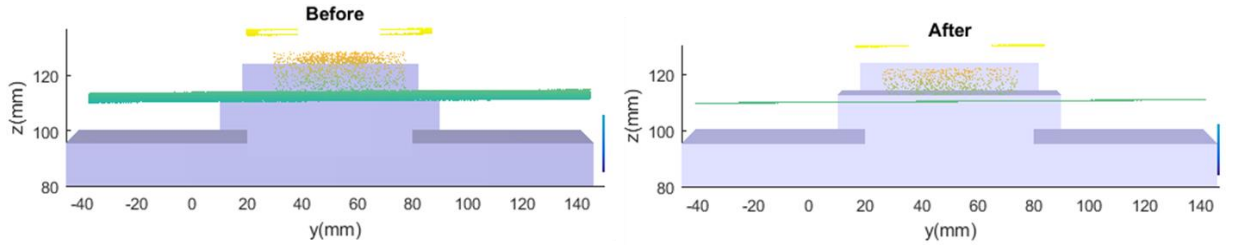


Figure 72 **(a). Front view of the seven point sets before T^* is applied; (b) Front view of the seven point sets after T^* is applied.**

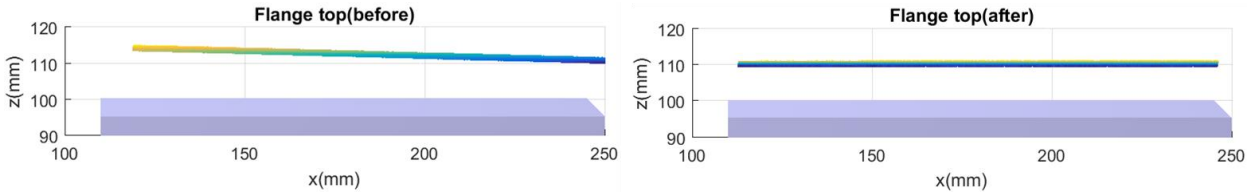


Figure 73 **(a). Side view of the point set that represents flange top before T^* is applied; (b) Side view of the point set that represents flange top after T^* is applied.**

Similarly, the point set for the top face of the neck must lie above the cutting line of the top face, which is already satisfied. Figure 74 shows how T^* compensates for the angular error about Y-axis and makes the point set approximately sit on the horizontal plane, z=130 mm.

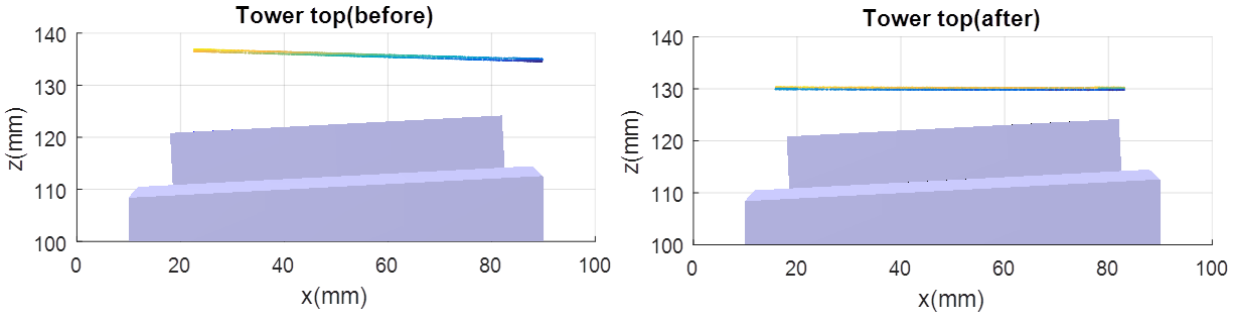


Figure 74 **(a). Side view of the point set that represents tower top before T^* is applied; (b) Side view of the point set that represents tower top after T^* is applied.**

Also, four virtual gauges, shown as the vertical lines in Figure 75 are deployed to check two walls' thickness. By applying T^*, the point sets moved to the new positions without touching the virtual gages. It shows that the requirements on thickness can be satisfied.

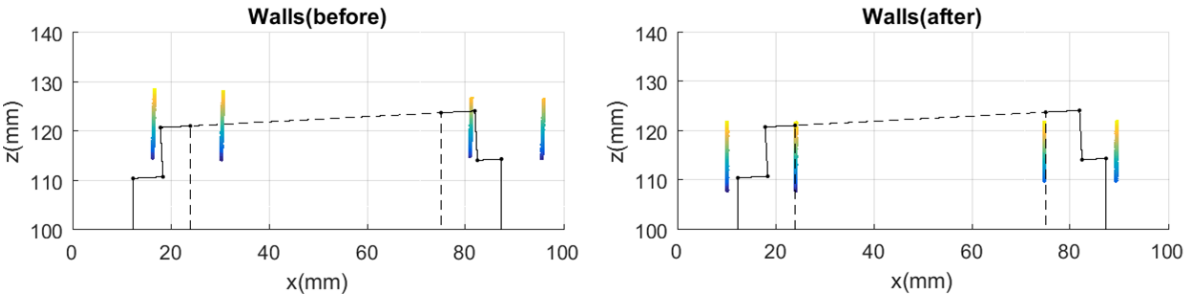


Figure 75 **(a). Side view of the point sets that represents four walls before T^* is applied; (b) Side view of the point sets that represents four walls after T^* is applied.**

The optimal HTM shown in Eq. (24) suggests how to adjust machining coordinate frame, which is done by modifying the machining paths in the G-code.

Then the HTM will be imported into the two methods that generate adaptive tool path.

Automated work offset modification

The offset calculation was first implemented on a small prototype with all but one face of the prototype has an inclination of θ = 3° around Y-axis. The picture of CAD files of designed small prototype, the to-be machined stock, and the table-fixture-part set-up in the machine CAD model is shown in Figure 76.

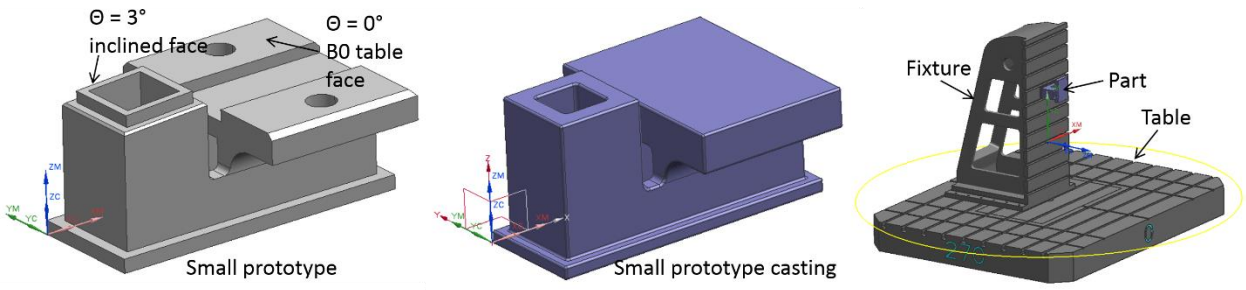


Figure 76 Scaled part CAD and the machining set-up

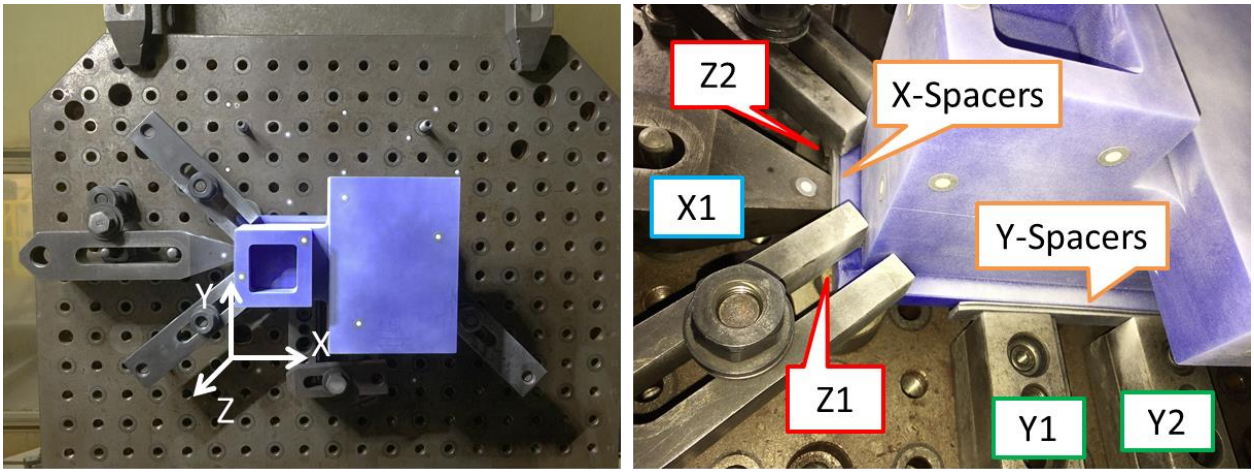


Figure 77 (a) Scaled model wax casting clamped on the machine table, (b) Spacers placed between locators and the casting to simulate the part transformation.

Table 20 provides the work offset values before and after compensation that are required to machine the part. The ideal work offsets before compensation are values obtained during the tool path generation in the CAD/CAM software and after touch probing the part location in the machine workspace by the operator. The part was designed to be machined at two different table rotation values i.e., B = 0° and B = 357° because of the 3° inclination face. The after compensation values were calculated based on the displacements provided by the virtual gage analysis matrix as shown below:

$$\begin{bmatrix} x_{cor} \\ y_{cor} \\ z_{cor} \end{bmatrix} = \begin{bmatrix} 3.033 \\ 4.438 \\ 2.072 \end{bmatrix} \quad \text{and} \quad b_{cor} = -2.452°$$

Table 20 Scaled part work offsets before and after compensation

	Before compensation			After compensation	
	G506	G507		G506	G507
X_{old}	-129.99	-105.572	X_{new}	-106.93	-82.177
Y_{old}	951.168	951.168	Y_{new}	955.606	955.606
Z_{old}	463.171	469.340	Z_{new}	470.249	475.201

The offset calculations and its NC program correction are shown in Figure 78. The software adds additional lines of offset calculation equations with controller variables in the NC program so that the implementation can be materialized in the controller. The left snippet shows the original NC program and the right snippet shows the modified NC program. The additional lines added

between line N50 and N55 can be observed in the modified NC program. The rest of the program remained the same.

```
N5 SET_WCS_2ROTAB(505,506,357.000)

; T3 3 INCH FACEMILL
N10 L140(1)
N15 L106(3)
N20 T11
N25 G0 G17 G40 G64 G90 G94 G500 M5
N30 M31=101
N35 M31=102
N40 M31=103
N45 M31=104
N50 M31=106
N55 G505
N60 SET_REACH(400.000)
N65 ; OPN: B0 MILL DATUM A FACE
N70 B=DC(0.0)
N75 G0 X237.16 Y106.76 Z224.326 S752 D01 M3 W0.0
N80 Z108.4
N85 G1 Y45.8 F631.7
N90 Y-145.8
N95 Y-187.71
N100 G0 Z224.326
N105 X187.63 Y106.76
N110 Z108.4
N115 G1 Y45.8
N120 Y-145.8
N125 Y-187.71
N130 G0 Z224.326
```

```
N5 SET_WCS_2ROTAB(505,506,357.000)

; T3 3 INCH FACEMILL
N10 L140(1)
N15 L106(3)
N20 T11
N25 G0 G17 G40 G64 G90 G94 G500 M5
N30 M31=101
N35 M31=102
N40 M31=103
N45 M31=104
N50 M31=106

;=============================================
;BEGINNING OF CODE ADDED FOR OFFSET COMPENSATION

DEF FRAME U_OLDOFFSET001
U_OLDOFFSET001=$P_UIFR[05]
DEF FRAME U_OLDOFFSET002
U_OLDOFFSET002=$P_UIFR[06]
DEF REAL U_TEMPW00X, U_TEMPW00Y, U_TEMPW00Z
U_TEMPW00X=(((3.2959+$P_UIFR[05,X,TR]+$P_UIFR[05,X,FI])*0.999450+(
U_TEMPW00Y=4.6611+$P_UIFR[05,Y,TR]+$P_UIFR[05,Y,FI]
U_TEMPW00Z=(((3.2959+$P_UIFR[05,X,TR]+$P_UIFR[05,X,FI])*-0.033159+
$P_UIFR[05]=CTRANS(X,U_TEMPW00X,Y,U_TEMPW00Y,Z,U_TEMPW00Z,B,-1.900
STOPRE
U_TEMPW00X=(((3.1710+$P_UIFR[06,X,TR]+$P_UIFR[06,X,FI])*0.999450+(
U_TEMPW00Y=4.6611+$P_UIFR[06,Y,TR]+$P_UIFR[06,Y,FI]
U_TEMPW00Z=(((3.1710+$P_UIFR[06,X,TR]+$P_UIFR[06,X,FI])*-0.033159+
$P_UIFR[06]=CTRANS(X,U_TEMPW00X,Y,U_TEMPW00Y,Z,U_TEMPW00Z,B,-1.900
STOPRE
M1

;END OF CODE ADDED FOR OFFSET COMPENSATION
;=============================================
```

Figure 78 Offset calculations implemented in the NC program. Left – original NC program and Right – corrected NC program with additional lines for offset compensation.

Vericut Simulation

The corrected NC program was later verified by performing a machining simulation in Vericut® software. A machine model is first constructed in the Vericut® space by importing the required CAD files of the machine components including table, fixture, and tools. The CAD files are then imported in to the virtual machine model in Vericut® software. Figure 79 shows the screenshots before and during the virtual machining process. Since a wax casting was utilized for scaled part implementation, the CAD file of the casting was considered as the stock file for virtual machining, as there were no significant differences between the scanned surfaces point cloud and actual surfaces because of the surface finish. However, the CAD file was transformed in the virtual space to reflect the real-world transformation of the wax part achieved by spacers as shown in Figure 3(b). The modified NC program was then loaded in to the software and used as input for virtual machining.

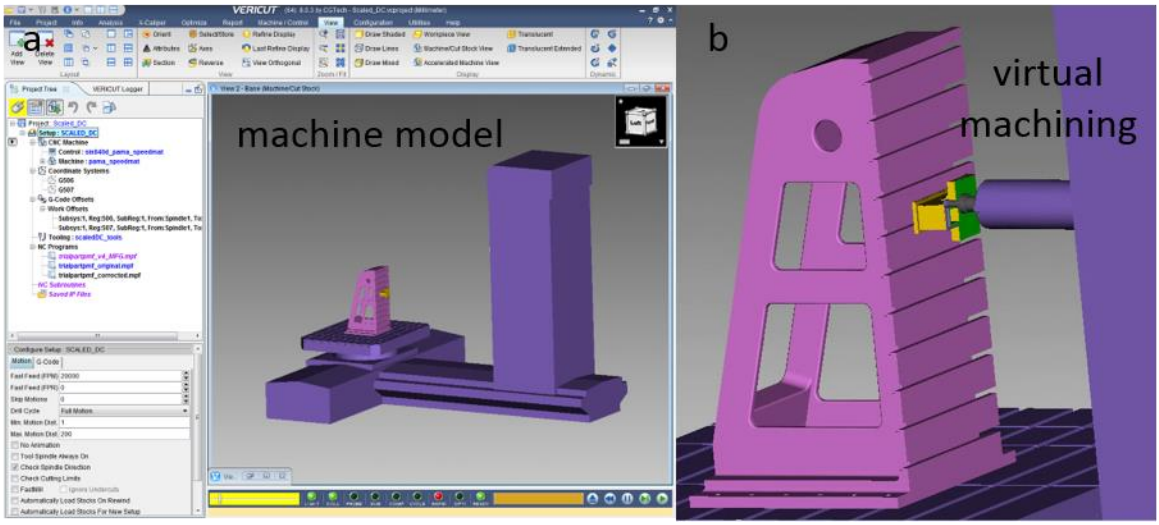


Figure 79 (a) Vericut® software view with the entire machine model set-up and the software UI, (b) During machining simulation using the corrected NC program.

Figure 80(a) shows the differences in surface profile of the original CAD file to that of machined file. Blue represents excess material and red indicates gouged material. It can be observed that at 1 mm tolerance (Figure 80(b)), there are no differences in surface profiles and all surfaces are within the defined tolerance values. Moreover, the part was moved in all 6 degrees of freedom whereas the compensation was only performed in X, Y, Z and B angles. It is evident that the shown differences in the 0.2 mm tolerance profiles are because of the uncompensated X and Z rotation of the part. This predicted result match the 3D measured report in Figure 82. Therefore, it has been demonstrated that the virtual machining using the machine model and Vericut® software provides reliable outcomes in estimating the surface profiles after machining with the corrected NC program.

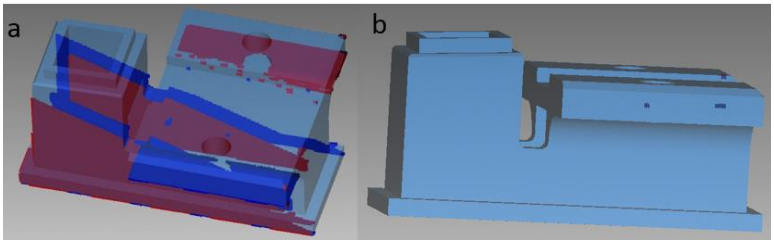


Figure 80 Analysis of the machined stock compared to the designed CAD file. (a) Highlighted differences at 0.2 mm tolerance values, (b) Highlighted differences at 0.2 mm tolerance values.

Regeneration of NC Program using Siemen's NX Journal

The journal method was implemented on a scaled part. As mentioned above, the requirements to implement the journal method in transforming the toolpaths and post processing the toolpath to G and M codes depends on the availability of the toolpaths in NX CAM module. As the original

NC program of the scaled part was defined and created using the Siemens NX software, the journal method was seamlessly tested on the scaled part. Figure 81 shows the screenshot of the NX software CAM module UI with the scaled part and the defined machining operations. The machining operations are highlighted on the CAD file. The shifted tool paths after executing the journal can be seen in the workspace. It has to be noted that the toolpaths are transformed instead of transforming the CAD file in this method. The transformed toolpaths are then post processed into G-code.

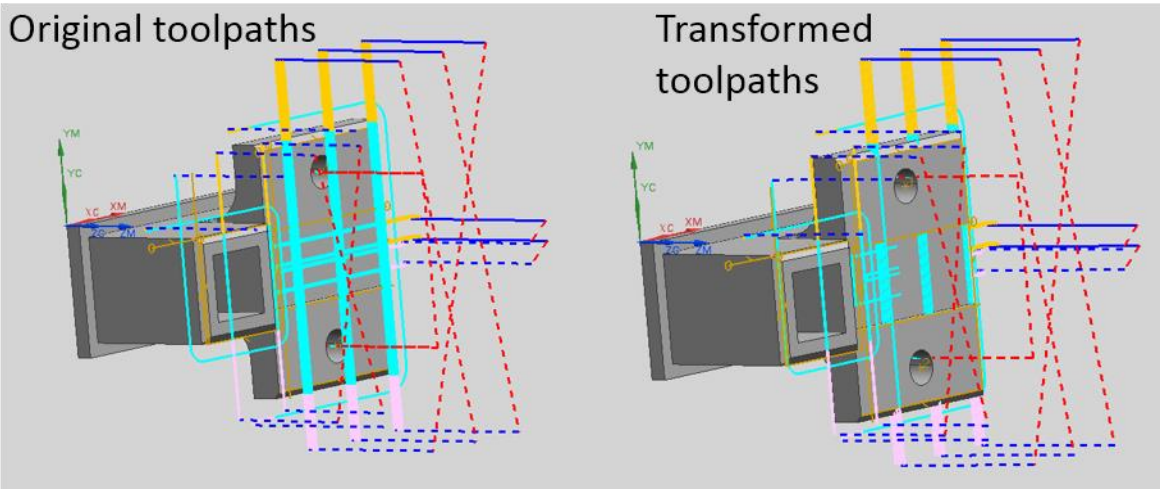


Figure 81 Toolpath transformation after executing the journal in Siemens NX CAD/CAM software. The journal later post processed the toolpaths into G-code based on the user selected post processor.

The developed system was validated experimentally by machining the part using the compensated G-code. The error map of the finished part was built by comparing the scan data and the nominal CAD model of the finished part. As shown in Figure 82, the translational and rotational offsets caused by the spacers are compensated. The feasibility of using virtual gage analysis to compensate workpiece variation is thus demonstrated.

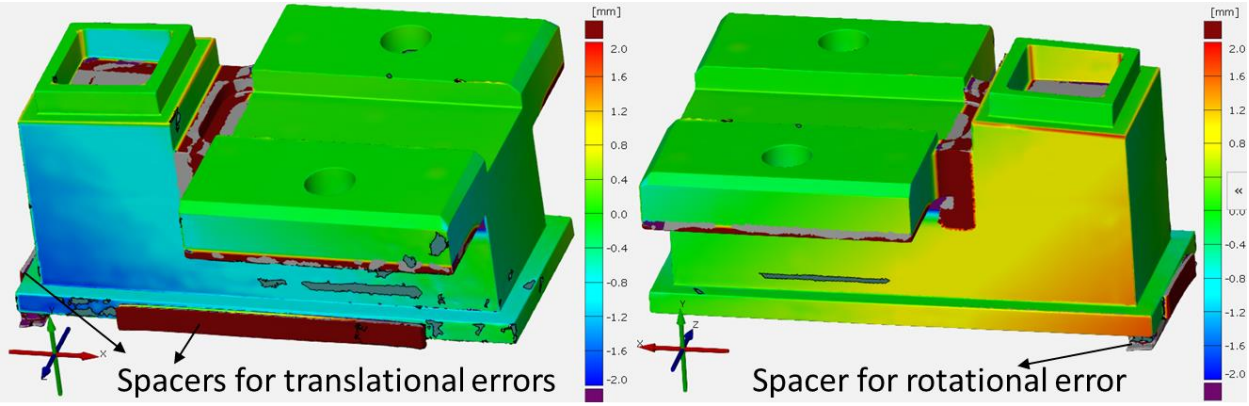


Figure 82 **The error map of the finished part.**

4.3.5 Inner wall thickness averaging using different target function

Even the finished part satisfies all GD&T requirements, there is still a room to improve the algorithm. In Eq. (25), all virtual gages are considered equally important, and their weighting values in target function are all 1. For different purposes of optimization, one can add more variables and use different weighting values for different virtual gages. For example, if the target function aims to average the minimum inner wall thicknesses by sacrificing other gages of J-groove, Eq. (25) can be modified to meet the requirement.

A simplified case of averaging two parallel wall thicknesses controlled by two virtual gages is shown in Figure 83. The point sets are only allowed to move horizontally and the minimum wall thicknesses are controlled by q_1 and q_2, the shortest distances from the point sets to the corresponding virtual gages (two dashed lines). Equal minimum thicknesses for these two walls can be achieved by minimizing the thickness of the thicker wall, $\max(q_1, q_2)$. In linear programming, we have,

$$\min_x q' \tag{98}$$

subject to $0 \le q_j \le e_{i,j}(x),\ q_j \le q' \forall i, j$, where $e_{i,j}$ is distance from the ith point in the jth point set to the jth virtual gage, q_j is the lower bound of $e_{i,j}$ and q' is an upper bound of q_j.

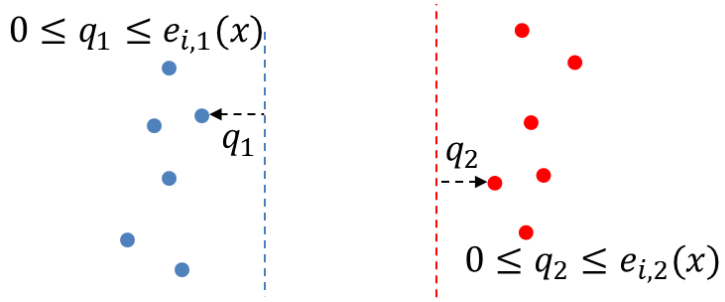


Figure 83 **Wall thickness equalization problem controlled by two virtual gages.**

The optimal solution is given by $q_{1,opt} = q_{2,opt} = q'_{opt}$ and average the wall thickness. The problem of wall thickness equalization can be generalized based on Eq. (25),

$$\min_{T\in\Omega}\left[\sum_{i=1}^{n}(g_i s_i + u) + \sum_{j=1}^{k} f'_j q'_j\right] such\ that$$
$$-u \le -s_j \le e_{i,j} \le q_i \forall\, r_{i,j} \in S_i, i = 1, \dots n, q_{2j-1}, q_{2j} < q'_j, j = 1, \dots k \tag{99}$$

where g_i is the non-negative weighting coefficient for the slack variable, s_i of the ith virtual gage, u is the largest slack variable, f'_j is the non-negative weighting coefficient for the jth wall

thickness pair to be equalized and q_j' is the upper bound for the distance measurements, q_{2j-1} and q_{2j}.

For the unscaled part, the minimum inner wall thicknesses of the neck, left wing and right wing are checked against 12 virtual gages using distance measurements, q_1~q_{12}. Therefore, six more variables, q'_1~q'_6 are introduced, and $k = 6$. f'_j is selected to be 50 by trail and error. The optimal solution of Eq. (28) expressed in HTM is given by,

$$T^* = \begin{bmatrix} 1 & 0 & -0.0026 & 0.49 \\ 0 & 1 & 0 & -5.22 \\ 0.0026 & 0 & 1 & -0.75 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (100)$$

The HTM shown in Eq. (29) compensates the displacements of the part and average the inner wall thicknesses as shown in Figure 84. Each pair of parallel inner walls is predicted and listed in Table 21. The inner walls are more evenly distributed. The results demonstrate the possibility of introducing additional variables and constraints in linear programming problem to achieve different optimal objectives.

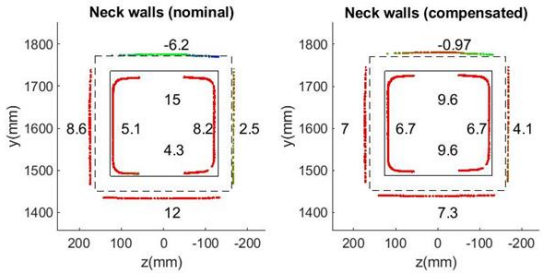


Figure 84 **Virtual gages, point sets and the machining allowance with wall thickness averaging (before and after compensation)**

Table 21 **Predicted and measured inner wall thickness (mm)**

Inner wall	Predicted (before)	Predicted (after)
Neck top	27.0	21.6
Neck bottom	16.3	21.6
Neck left	17.1	18.7
Neck right	20.2	18.7

4.3.6 Software and System Requirements

A CAD software (e.g. AutoCAD or ProE) is required for parameterization and visualization of the virtual gages. MATLAB with Computer Vision System Toolbox is necessary for running two programs, which are broken down into functions listed below.

The implementation of modified NC program on the CNC machine requires a programmable CNC controller. In this project demonstration, Siemens Sinumerik 840 series controller was used. Sinumerik controller variables will be embedded in the final modified NC program to provide an operator hands-free approach to machine the part. Depending on the particular configuration used to modify the NC program, Siemens NX (v10.0) could also be required.

To run the software, it must be compiled on a desktop or laptop PC. The software's are written in a combination of programming languages and CAD-related software. For the NC program modification software, program was created using a proprietary programming language (MATLAB vR2016b). For the machining simulation, the system utilizes Vericut software and requires a defined CNC machine model with tool definitions and CAD files.

4.3.6.1 List of the functions and Scripts

1. EditScanPath.m

The MATLAB script used to process the scan path waypoint list generated via the Vericut MDI includes the following parameters that should be set. Outputs are four files: a Position file and an NC program file for both discrete and velocity mode scanning

a. filepath: This is the file path to the folder containing the waypoint text file. Leave as empty if the matlab script is in the same folder.
b. filename: This is the name of the waypoint text file generated with Vericut (excluding the .txt extension). This is used as the base name for the other files generated by default, with differing endings added to each of the four files generated.
c. on: This is a string specifying the M code used to turn on the trigger signal.
d. off: This is a string specifying the M code used to turn off the trigger signal.
e. speed: This is the desired feed rate of the machine while scanning in mm/s. For the testing done, this was set to 100 mm/s since this, together with the capture frequency of 1000Hz used resulted in scan line spacing of 100 microns, equal to the spacing of the sample points along the scan line.
f. freq: This is the desired scan sample frequency in Hz. Together with the machine speed, this determines the spacing between scan line samples. For the scanner model used, a frequency of 1000 Hz was found to be ideal, with larger frequencies

filling the buffer and requiring additional processing time, resulting in a total capture time similar to that needed using 1000Hz to capture the same input.

g. batch: This is number of samples in each scan batch. This should be based on the longest scan pass length + acceleration distance in combination with the scan sample spacing. This has a minimum of 50 and a maximum of 15,000.
h. buffer: This is the nominal buffer length in mm added to the ends of each pass to account for the acceleration and deceleration period as well as startup delay to ensure that the desired scan path segment is traversed while at constant velocity and while the scanner is measuring. Experimentation on the PAMA indicated that the machine took around 0.3 seconds to reach the desired constant velocity of 100mm/s, so a buffer of 30mm would be sufficient for the acceleration, but given the significant variation in startup delay this was increased to 50mm to ensure no loss of data in the desired range.
i. maxscanlength: This is the length of the longest scan segment in mm. Since scan batch sizes cannot be changed during execution of the program, this adds a pause in machine movement after shorter scan segments to ensure that the previous batch is complete before starting the next pass. Ideally, scan segments should be designed to all have similar length to reduce wasted time.
j. discspacing: This is the desired spacing in mm of the measurements when in discrete mode. There need to be enough measurements in each scan pass to result in unique alignment with the continuous data, so this will depend on the part size and uniqueness of features on the surface. For tests on the PAMA and the scaled part, a spacing of 10mm was found to work well.
k. scannertransf: This is a 4x4 transformation matrix that gives the position and orientation of the scanner measurement frame with respect to the tool frame of the machine (end of spindle). For calibration scans, the value of this matrix does not matter as the raw data is used. After calibration, the transformation matrix identified should be copied here.
l. headercode: This is a text string including any NC code necessary at the beginning of the program to configure the machine.
m. footercode: This is a text string including any NC code necessary at the end of the program to configure the machine.
n. noscan: This is a string that specifies the keyword used to indicate collision avoidance waypoints in the scan path waypoint list generated in Vericut.
o. areascan: This is a string that specifies the keyword used to indicate a three point specification of a rectangular area to be scanned in the scan path waypoint list generated in Vericut.
p. defaultwidth: This is the maximum separation in mm desired between parallel passes used to cover rectangular areas specified in the scan path waypoint list. If the width of the rectangular area does not divide evenly by the default width specified here, then the number of passes is rounded up and scan passes are

distributed evenly resulting in a slightly smaller separation. A default width of 50mm was used in testing to ensure that parallel passes maintained at least some overlap regardless of part depth since the width of the scanner beam at the near end of the measurement range is 51mm (increasing to 73mm at the far end of the measurement range).

2. ProcessCalibrationScans.m

The MATLAB script used to process the calibration scans and calculate the transformation matrix between the machine end effector and scanner frame allows for the following input:

a. path: This is a text string giving the file path to the raw scan files generated by the scanning executable. The base name of the files themselves should be OUT unless the files or the code to generate them has been modified.
b. hybridmode: This is set to 1 if using hybrid mode scanning and will then look for an offset file describing the offsets calculated between the continuous and discrete scans as generated by the data alignment script. Set to 0 if using a machine which has consistent startup timing and therefore a constant acceleration offset, or if using an encoder signal to trigger scanning.
c. nums: This is a list of the file numbers of the calibration scan passes. If using the basic set of variations described above, this should be the numbers 0-4. If additional variations are used to increase accuracy, then the scan file numbers should be edited appropriately – otherwise this need not be changed.
d. startpos: This should be a matrix with rows corresponding to the start position [X Y Z W B] axis locations for each scan pass.
e. step: This is the distance step between scan samples in mm.
f. offsetpath/offset: If hybrid mode is being used, then specify the file path to the offset file which was generated by the data alignment script. If an encoder signal is being used to trigger scans, simply set the offset to 0. If velocity mode is being used, then enter the correct offset value for the machine found by scanning the same scan path segment on a sloped surface from opposite directions, plotting the raw results (reverse the order of one of the scans so that the displayed results are again in the same direction) of height with respect to sample number, and identifying the offset (in number of samples) between respective sample numbers for the same part height value. The correct offset will be half the total difference since each scan is offset in opposite directions.

3. FindOffsets.m

This script processes scans from both velocity mode and discrete mode scanning and calculates the offset of each velocity mode segment with respect to the actual positions recorded by the sparse discrete data. The following inputs are needed:

a. path: This is a string that contains the entire file path and the “base” name of the continuous scan displacement files. The “base” name is the portion of the file

name before the #.txt portion. The code increments through all displacement files by appending the #.txt portion during operation.

b. nums: This is a row vector that corresponds to the number of scan passes. It should contain the numbers 0 to N-1 where N is the number of scan passes.
c. discretepath: This is a string that contains the entire file path and file name for the discrete displacement measurements.
d. savepath: This is a string that contains the entire file path and file name for the outputted file containing the offset for each of the scan passes.
e. discretenums: This is a cell array that separates the discrete scan lines into the corresponding scan passes of the continuous measurements. The number of cells in the cell array corresponds to the number of scan passes for the continuous measurements, and the i+1 cell contains all discrete measurements for the i scan pass.

4. ***string = point2string (point) or point2string(point,B)***

This function converts a vector containing the numerical axis positions [X,Y,Z,B,W] into a string containing the NC code representation of that point. i.e. [0 0 0 0 0] is converted to 'X0 Y0 Z0 B=DC(0) W0'. The default is to use the B=DC(#) format for specifying B axis commands, but if there are two arguments passed to the function then it will instead use the B# format.

5. ***done = writescanline(file,posfile, start, finish, on, off,buffer,setnum, spindle,increment)***

This adds the relevant entries for one continuous scan segment to the NC code file and the position file specified by the inputs 'file' and 'posfile'. 'start' and 'finish' are the axis commands for the start and end of the desired scan segment. 'on' and 'off' are strings representing the M command used to turn the trigger signal on and off. 'buffer' is the length of the extension added to each end of the scan path to account for acceleration and startup delay. 'setnum' is the path segment number. 'spindle' is the angle of the spindle. 'increment' is the step size between scan samples. The output is a dummy variable and is returned 1 on completion.

6. ***newsetnum = writediscscanline(file,posfile, start, finish, on, off,setnum, spindle,spacing)***

This adds the relevant entries for one scan segment done in discrete mode to the NC code file and the position file. Inputs are equivalent to those described in the function 'writescanline'. The output is the value of the input 'setnum' incremented by one.

7. ***[starts finishes] = dissectpath(start, finish, batchlength, buffer)***

This function checks if a scan path segment is too long to be completed in one scan batch and if so, breaks it into multiple segments. 'start' and 'finish' are the endpoints of the scan path segment. 'batchlength' is the length possible to scan in one batch given current batch size, feed rate, and sample rate settings. 'buffer' is the length of the extension necessary at either end of the scan segment to allow for acceleration/deceleration and startup delay. The output is two

matrices where each row contains the start or finish axis positions for the resulting segments. If the segment is not too long for a single batch then the outputs will be a single row each and identical to the inputs 'start' and 'finish'.

8. offset = CalcOffset(vel,disc,discloc,searchnum)

Finds the offset value for a single scan pass. 'vel' is the data gathered in velocity mode. 'disc' is the data gathered in discrete mode. 'discloc' is the nominal sample numbers from the velocity data that the discrete data should align with if there were no offset. 'searchnum' is the distance in number of samples that the offset should be searched for (i.e. the maximum offset allowed as a result). The output is the offset measured in number of velocity data samples. This can be a fractional value if the correct offset lies between two samples.

9. pts = calibplane2pts(I,t)

Extracts the location of the corner points from the raw scanner data based on user identification of the flat areas of each plane. 'I' is the 'image' generated from the scan data and 't' is the threshold used for plane fitting. The output is a 3x4 matrix of the 4 corner points extracted as measured in the scanner frame (X and Y values are given in terms of sample number. Z is the actual measured distance to the surface from the scanner).

10. [R t transf pt] = FindScannerTool(meas, cmm, varargin)

Calculates the transform as well as calibration corner points. 'meas' are the locations of the corner points as measured by the scanner. It is a 3 by p*n matrix where p is the number of keypoints (4 – the corners between the 5 surfaces on the calibration part) and n is the number of variations in start points for the scan passes (5 for the minimal set of passes to identify the transformation). The X and Z values are derived directly from the scanner measurements. The Y value of meas is based on the distance that the machine has traveled from the starting position when that point was passed. 'cmm' are the axis locations for the start of each scan pass variation. 'varargin' is an optional argument that can be used to specify the location of the keypoints instead of calculating them from the scanner measurements. This was evaluated as an alternate calibration strategy, but was found to be less accurate if the locations of the corners were determined via touch probe measurements of the surfaces. The outputs are the rotation matrix 'R' describing the orientation of the scanner with respect to the machine tool frame, the translation 't' of the scanner with respect to the machine tool frame, 'transf', which is the full 4x4 transformation matrix that combines 'R' and 't', and 'pt' which are the locations of the keypoints that were identified (corners of the calibration block) as measured in the machine base frame. The forward kinematics function used for the machine is hardcoded into this function, so if a different machine is used, an alternate forward kinematics function should be created and the reference in this code changed.

11. RANSAC plane fitting code (ransacfitplane, ransac, fitplane, iscolinear)

This is public MATLAB code that implements the RANSAC (Random Sample and Consensus) plane fitting method (a relatively fast method for fitting planes to data while automatically rejecting outlier data from influencing the fit) and is available without use restrictions (other than maintaining the reference to the author in the code comments) from the MATLAB file exchange. The code was developed by Peter Kovesi of The University of Western Australia.

12. NEWxyzPoints = *PC_HTM*(xyzPoints,HTM)

Returns a new point set data by applying a homogeneous transformation matrix, HTM to a 3-D point data set represented by a matrix with three columns.

13. ptCloud = *pointCloud*(xyzPoints)

Stores a 3-D point data set, xyzPoints as a point cloud object, ptCloud. The point cloud object can be used in the following point set calculation. See https://www.mathworks.com/help/vision/ref/pointcloud-class.html?searchHighlight=pointCloud&s_tid=doc_srchtitle for more details. The point set can be retrieved from the point cloud object by accessing its "Location" property — i.e., ptCloud.Location.

14. indices = *findPointsInROI*(ptCloud,roi)

Searches a point cloud object, ptCloud and return the indices of the points which are inside of the region of interest, roi. For more information, see

https://www.mathworks.com/help/vision/ref/pointcloud.findpointsinroi.html?searchHighlight=findPointsInROI&s_tid=doc_srchtitle

15. ptCloudOut = *select*(ptCloud,indices)

Returns a pointCloud object containing the points selected using linear indices.

https://www.mathworks.com/help/vision/ref/pointcloud.select.html?searchHighlight=select&s_tid=doc_srchtitle

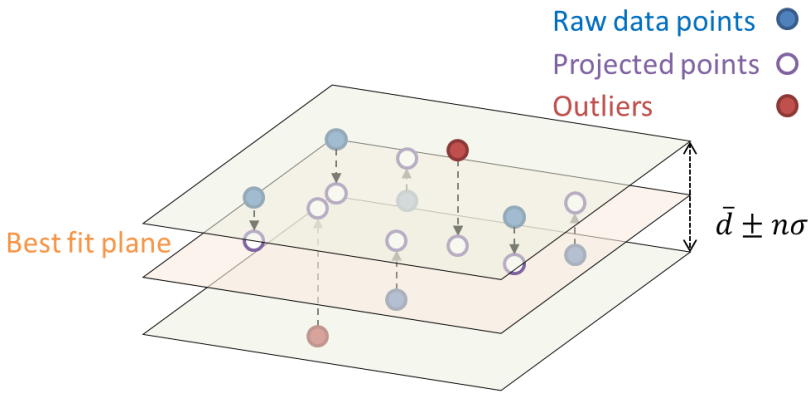
16. [model ,inlierIndices,outlierIndices]= *pcfitplane*(ptCloudIn,maxDistance,referenceVector,maxAngularDistance)

Detect a planel and extract it from the point cloud, ptCloudIn. This function provides a robust way for data sampling and feature extraction. More details can be found at:

https://www.mathworks.com/help/vision/ref/pcfitplane.html?searchHighlight=pcfitplane&s_tid=doc_srchtitle#busqbp7-1-maxDistance

17. NEWxyzPoints = *LSQ_plane_filter*(xyzPoints,n)

As shown in Figure 85, the function fits the input point set as a best fit plane and remove the outliers in the point set, which deviate by n times the standard deviation(of the distance to the best fit plane). The recommended value for n is 2.5~3 (remove about 0.3~1.6% of points).



d_i: the distance from the i[th] point to the best fitting plane
$\bar{d}$: mean of the d_i's
σ: standard deviation of the d_i's

Figure 85 **Outlier removal using the point-set filter.**

18. $[A_n, b_n]$=*LPP_Constraints_4DOF*(xyzPoints,p,n,N)

This function generates the n[th] constraint set, $A_n v \leq b_n$ for the linear programming problem (slack variables are included) with a total of N virtual gages ($N \geq n$), where p is the row vector with four plane parameters, representing the gage plane, $ax + by + cz + d \geq 0$.

19. x = *linprog*(f,A,b)

Solves the solution of the linear programming problem specified by:

$\underset{v}{\text{Min}}\, f^T v$ such that $Av \leq b$,

where f is the vector of coefficients of target function, v is the vector of all controllable variables, $b = \left[{b_1}^T \ \ldots \ {b_N}^T\right]^T$ is the vector of linear inequality constraints and $A = \left[{A_1}^T \ \ldots \ {A_N}^T\right]^T$ is the matrix of linear inequality constraints. For more information, see

https://www.mathworks.com/help/optim/ug/linprog.html?searchHighlight=linprog&s_tid=doc_srchtitle#buusznx-A

20. [C_PS,C_PS_HTM,PC_HTM,tol_PS,tol_PS_HTM]=*color_tol*(xyzPoints,HTM,p)

Returns the color code of every point in the point cloud based on the distance from the point to the virtual gage, where

C_PS is the color code matrix of the uncompensated point set,

C_PS_HTM is the color code matrix of the compensated point set (after the compensation homogeneous transformation matrix has been applied),

PC_HTM is the point set with compensation,

tol_PS is the machining allowance of the uncompensated point set,

tol_PS_HTM is the machining allowance of the compensated point set,

HTM is the compensation homogeneous transformation matrix and

p is the row vector with four plane parameters, representing the gage plane, $ax + by + cz + d \geq 0$.

21. *pcshow*(xyzPoints) or *pcshow*(ptCloud)

Displays points set data or point cloud.

See https://www.mathworks.com/help/vision/ref/pcshow.html for more detailed information.

4.3.6.2 Data structure

For development of laser scan paths for in machine scanning, there are three different file types that are generated: path waypoint lists, NC code, and position files. The NC code is an .MPF file generated that is suitable for loading and running on the machine. The path waypoint list and position files are structured files which are used by the path generation and scanning code and have the following format:

1. Path waypoint list

The path waypoint list is created using the Vericut MDI and is saved as a text file. Each row of text should correspond to either the axis coordinates for a waypoint, a keyword to modify processing of following waypoints, or a spindle repositioning command of the form SPOS=[ang]. Each waypoint is listed as a row of text with the machine coordinates for that waypoint, with each axis specified by a letter followed by the numerical coordinate. Spaces should separate each axis. Waypoints should generally be listed in pairs corresponding to the beginning and end of each scan segment. Exceptions to this are with respect to commands that follow a defined keyword such as ‘noscan’ (for collision avoidance waypoints) which is followed by a single waypoint, or ‘areascan’ (for scanning rectangular areas) which is followed by three waypoints.

2. Position file

The position file is a CSV file which is automatically generated by the code which processes the path waypoint lists and generates the NC code to be run by the machine while scanning. It contains the transformation matrix representing the position and orientation of the scanner frame with respect to the machine tool end effector frame, the number of invalid scan samples expected to be acquired during the acceleration period, and the starting axis coordinates and step difference between samples, and number of samples for each scan segment. A sample of

the format is given below, including the first two scan segments (note that the transformation entered in this sample is simply the identity matrix – in actual practice this would be replaced by an actual transformation representing the location and orientation of the scanner):

#Scanner Transformation					
1	0	0	0		
0	1	0	0		
0	0	1	0		
0	0	0	1		
#Acceleration scanlines					
500					
#Set 1					
*Start Location					
-535	969.63	1126	-500	90	0
*Step					
0	0.1	0	0	0	0
*Number of Steps					
2401					
#Set 2					
*Start Location					
20	1225.63	1708	-635	0	0
*Step					
0	-0.1	0	0	0	0
*Number of Steps					
2501					

The raw data output by the in-machine scanning code is in the form of a text document with the numerical data measured by the scanner with a header row giving the location along the scan line of each column. (-40mm to 39.9mm every 0.1mm) The raw values are integers which correspond to measurements in mm x 10^{-5}. The values themselves correspond to the distance to the part (or a very large negative number in cases of no valid surface detected) and each row is one scan line sample in the order that they were acquired.

When using the hybrid method for in-machine scanning (necessary for the PAMA machine) an offset file is also generated to record the offset of the continuous version of each scan segment from the actual positions measured using the discrete method. This is in the form of a vector with the offsets for each scan segment in sequence and is saved as a text file using a space delimiter for transfer between the MATLAB and C++ components.

Since the programs are developed in MATLAB. The data structure follows the conventions of MATLAB.

1. Point set data(matrix form)

A point set is represented by a k by 3 matrix, where k is the size of the point set. The point set can be converted to point cloud class using “pointCloud” command. For the functions 1, 2, 6, 7 and 9 listed previously, the user should input a point set using a matrix.

2. Point clouds

The point cloud class (as shown in second function in the list of functions) is a MATLAB data structure for use of certain point cloud algorithms (functions 3, 4 and 5). It must be noted that point cloud cannot be fed into functions 1,2,6,7 and 9 directly. To input a point set data from a point cloud, one must extract the location information in a point cloud using its “Location” property—i.e., ptCloud.Location returns the matrix form of the point cloud data.

3. Virtual gage plane parameter sets

This structure is used in function 7 and 9. A virtual gage plane, which represents an inequality, $\mathrm{ax} + \mathrm{by} + \mathrm{cz} + \mathrm{d} \geq 0$ is specified using a row vector with four plane parameters, [a b c d], where $a^2 + b^2 + c^2 = 1$.

4. Homogeneous transformation matrices (HTMs)

HTM used in the programs and functions should followed the convention:

$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, where $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ is a rotation matrix and $[p_1\ p_2\ p_3]^T$ is a vector of translation.

4.3.7 Features & Attributes

The Adaptive machining implementation has the following features:

1. It is application agnostic. It can be used for raw casting metrology or finished (part) metrology.
2. It can be used with on-machine scanning or off-machine scanning
3. In-machine scanning can be used with a variety of trigger methods (encoder, time-based, discrete, hybrid) depending on the accessibility of machine signals and consistency of motion timing.
4. It supports both in-fixture scanning as well as scanning of the part and fixture separately
5. In its current implementation it support primarily planar surface metrology. Extensions to cylindrical surface metrology have been developed, but not tested on industrial examples.
6. In its current implementation, it has been used to verify dimensional/material condition tolerance specification. However, only supports form tolerance specifications.

7. The system enables quick and efficient manner to modify the nominal NC program and verify the compensation in workspace errors and collisions during machining, if any. The system does not affect the existing or saved work offsets, if any, by the machine operator.

4.3.8 Modes of Operation

The Technology is designed for new product introduction or current product improvements. Once the features are identified,

1. During a new part introduction, a metrology/manufacturing engineer sets up the Adaptive Machining criteria from the CAD model of the part.
2. Once set up, during production, the machine operator or production inspector scans the part.
3. For in-machine scanning approach, metrology/manufacturing engineer generates scan path from Vericut with automated in-machine scanning approach.
4. metrology/manufacturing feeds the data into the software, calling program 1 and then program 2 sequentially.
5. Then, offset information is imported to compensated NV program generator.
6. Operator load the compensated NC program into the controller and run the program.

5. ACCESSING THE TECHNOLOGY

BIP and IP claims have been documented in a separate file - “Post Project BIP and IP Claims”. Also, the technology and system requirements have been documented in the previous sessions - “Software and System Requirements”.

6. INDUSTRY IMPACT & POTENTIAL

These technologies can serve any manufacturing company that experiences problems related to setup errors, asset accuracy estimation, or incoming material variation. Specifically, industries where single rough piece-part cost can be in the tens of thousands of dollars, a single scrap piece (or even a piece that requires significant amount of rework) can easily wipe out any profit for a given time period. In large component manufacturing, the scrap cost is high, but the supply chain lead time tends to be orders of magnitude longer than the final machining process time. In these cases, there is simply no part available that can replace the scrap piece, creating delays in ship dates from that point down the remaining supply chain. This technology is proposing to minimize the exposure of machining facilities to the deviations that most frequently cause the conditions that generate scrap and/or rework.

7. TECH TRANSITION PLAN & COMMERCIALIZATION

Based on the membership level, some DMDII members have access to the final report as well as the user manuals of hardware and software that have been documented. Also, video demo has been recorded to show potential users the operation of the software.

Identify Future Plans

Software code developed from this project need to be maintained and improved for large implementation. DMDII, Caterpillar Inc. and IMVM team are working with commercialization partners on commercializing the current technologies for broader industry costumers as well as larger implementations inside Caterpillar Inc.

Identified Barriers to Adoption

The cost, both capital cost and variable cost, could be the barrier to prevent adoption of these technologies. Based on different scenarios, value calculation should be performed to justify the initial investment.

8. WORKFORCE DEVELOPMENT

There are more than 10 graduate students have been working on these projects as a cross functional team. Students built up their technical expertise and shared with other team members. By working in a production environment, they also built up more practical skills from solving real industry issues. Two conference papers and one journal paper have been published from the team.

9. CONCLUSIONS/RECOMMENDATIONS

From this Integrated Manufacturing Variation Management (IMVM) project, two technologies have been developed.

- Manufacturing asset volumetric error compensation (VEC) technology with laser tracker
- Adaptive machining technology with part scan and automatic planning for numerical control programming

Volumetric error compensation (VEC) is the use of a laser tracker to obtain machine errors. Machine tools normally are calibrated using classical methods with laser interferometers, dial gauges, and ball bars. These methods are slow and highly dependent on maintenance training and skill. They also provide a small view of the workspace errors, typically errors along a line, rather than throughout the volume. In contrast, the VEC methods utilize a laser tracker that can rapidly measure the entire volume by tracking and measuring the 3D location of the tool tip. The improvement of machine tool accuracy is more than 80% compared to uncompensated machine tool assets. Compared to B5.54 test, the error of this technology is less than 20%. Also, the measurement cycle is less than one shift.

Adaptive machining technology is compatible with either in-machine or off machine scanning approaches and be able to generate an optimized offset by fitting the scanning points cloud to the CAD model. Adaptive machining can be achieved by compensated tool path. It allows for

large reductions in setup times for new parts, new fixtures, or parts that see a large variation in the rough condition as delivered to the machining operation while minimizing human interaction in the machining setup process.

10. LESSONS LEARNED

Couple of lessons that learned from this project.

1. Thermal qusi-static error can contribute more than 50% in certain circumstance.
2. In future thermal deformation measurements of the machine tool, measurement of the machine temperature, rather than air temperature, is recommended.
3. Involve the commercialization partner into the project at the beginning of the project;
4. Keep the 1st adopter in technology development at an early stage;

11. APPENDICES

1. Video demo has been submitted to DMDII
2. Manual of Volumetric Error Compensation and Adaptive Machining attached below.